GraphGT: Machine Learning Datasets for Graph Generation and Transformation

Yuanqi Du[1]*, **Shiyu Wang**[2]*, **Xiaojie Guo**[5], **Hengning Cao**[1], **Shujie Hu**[2], **Junji Jiang**[3], **Aishwarya Varala**[1], **Abhinav Angirekula**[4], **Liang Zhao**[2]†
[1]George Mason University, [2]Emory University, [3]Tianjin University, [4] Thomas Jefferson High School, [5]JD.COM Silicon Valley Research Center
contact: ydu6@gmu.edu, liang.zhao@emory.edu

Abstract

Graph generation has shown great potential in applications like network design and mobility synthesis and is one of the fastest-growing domains in machine learning for graphs. Despite the success of graph generation, the corresponding real-world datasets are few and limited to areas such as molecules and citation networks. To fill the gap, we introduce GraphGT, a large dataset collection for graph generation and transformation problem, 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 based on research tasks, graph types, and application domains. We have significantly (re)processed all the data from different domains to fit the unified framework of graph generation and transformation problems. In addition, GraphGT provides an easy-to-use graph generation pipeline that simplifies the process for graph data loading, experimental setup and model evaluation. Finally, we compare the performance of popular graph generative models in 16 graph generation and 17 graph transformation datasets, showing the great power of GraphGT in differentiating and evaluating model capabilities and drawbacks. GraphGT has been regularly updated and welcomes 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, 5, 6], which aims at encoding graph structural information into a (low-dimensional) vector space, and 2) graph generation [7, 8], which reversely aims at constructing a 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 [9], graph convolutional network (GCN) [10], and graph attention networks (GAT) [11] have been proposed for various tasks including node classification [12], link prediction [13, 14, 15], clustering [2, 4] and others [16, 17].

Beyond graph representation learning, graph generation and transformation via machine learning start 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

*Equal Contribution
†Corresponding Author

Submitted to the 35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks. Do not distribute.

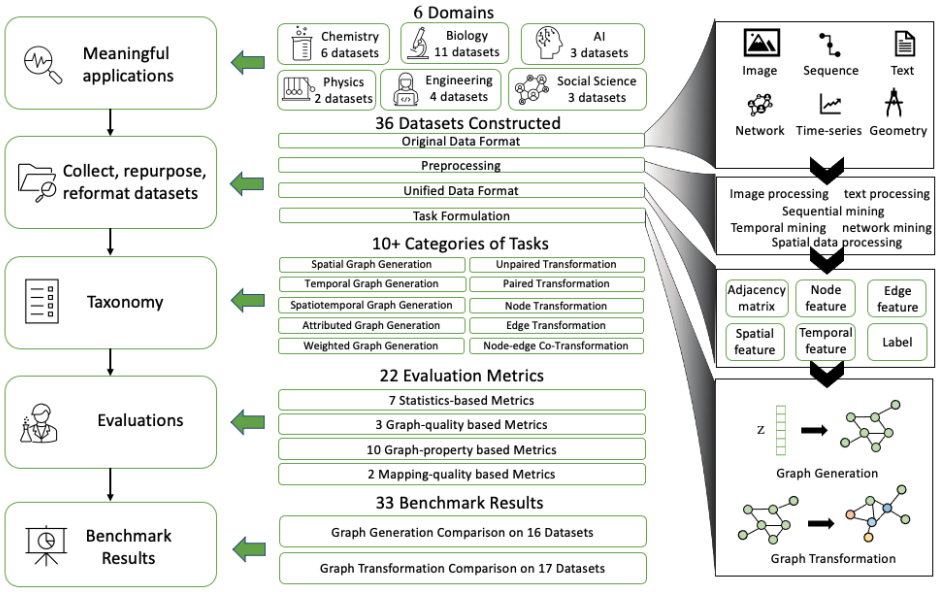

Figure 1: GraphGT dataset collection overview.

which require strong human prior knowledge and hand-crafted rules. Hence, graph generation and transformation via machine learning has 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 few of them have been studied targeting specific domains, such as GraphVAE [18], MolGAN [19] and JT-VAE [20].

Unlike graph representation learning which enjoys various benchmark datasets such as CORA, CITESEER and PUBMED for node classification [21], OAG for link prediction [22], and Molecule-LENET for graph-level prediction [23], graph generation and transformation via machine learning is still in its nascent stage and lacks comprehensive benchmark datasets that well cover different key real-world applications and types of graph patterns. Existing datasets are basically limited to few domains such as citation networks and molecules [7, 24]. Such data scarcity issue further leads to the following bottlenecks for the advancement of this fast-growing domain of graph generation and transformation: **(1) Difficulty in formulation:** graph structured data is complex in its nature; and the raw data in different domains may requires greatly different procedures to process or re-process in order to fit into a unified format. **(2) Limited number of application domains:** Although graph generation and transformation is a very broad generic concept that covers graphs in areas ranging from geography to biology, to physics, to sociology, to engineering, existing datasets only cover limited domains which prevents the development of graph generative models as well as applications in more diverse domains. **(3) Lack of taxonomy:** As the area of graph generation and transformation grows, the research tasks are diversified and hence require a well-defined categorization in order to have the dataset under the right category for the evaluation of the corresponding task. **(4) Lack of unified evaluation procedures:** the evaluation metrics used in existing research works are quite diverse and a gold standard for the evaluation procedure and metrics is needed. Moreover, the scarcity of existing datasets may bias the selection of elevation metrics to fit the limited number of existing datasets (e.g., molecules) but may not be general to other datasets. **(5) Lack of comprehensive model comparisons:** existing models are usually evaluated in a small number of datasets in very focused domains and some may be prone to "overfitting" to these datasets already, which significantly challenges the differentiation, evaluation, and advancement of the existing methods.

To tackle the aforementioned challenges, we introduce GraphGT, a large dataset collection for graph generation and transforamtion in machine learning, which **(1)** collects, re-purposes, re-formats a large amount of graph datasets, that **(2)** covers a variety of domains and subjects, **(3)** provides a systematic reviews and classifications of the datasets, **(4)** standardizes on the model evaluation procedures, and **(5)** provides benchmark results on a large amount of datasets. 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, we collect and construct CollabNet dataset and 7 brain network datasets from scratch for graph generation and transformation. Another 8 datasets are re-purposed by us from other applications into graph generation and transformation tasks for the first time. The remaining are from very different domains that share quite different terminology, formats, and data structures, which are reformatted by us to a unified format for the first time for easy access and use in a standardized manner.
- We provide and analyze results of graph generation on 16 datasets and graph transformation on 17 datasets using popular graph generation and transformation models. We observed that the performance of the comparison methods in different datasets (e.g., with different graph sizes, feature types, etc.) in different domains can be quite diverse. Hence GraphGT can be very helpful in differentiating the comparison methods, locating their drawbacks, and further advancing them.
- 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 Appendix E. In addition to the access via the Python API, GraphGT is open-sourced and available for download via GitHub at `https://graphgt.github.io/`.

2 Related Works

As graph representation learning enjoys an explosive growth in machine learning, numerous research works such as DeepWalk [9], graph convolutional network (GCN) [10], and graph attention networks (GAT) [11] have been proposed for various tasks including node classification [12], link prediction [13, 14] and clustering [2, 4]. Along with this, some datasets are proposed, such as datasets for node classification (CORA, CITESEER and PUBMED) [21], datasets for link prediction (OAG) [22], and datasets for Graph-level prediction (Molecule-LENET) [25]. To summarize and standardize these datasets, many data collections for graph representation learning has been proposed. Stanford Network Analysis Platform (SNAP) is a general purpose network analysis and graph mining library which contains massive networks with hundreds of millions of nodes, and billions of edges [26]. OPEN GRAPH BENCHMARK (OGB) is a diverse set of challenging and realistic benchmark datasets to facilitate scalable, robust, and reproducible graph machine learning (ML) research [27]. However, 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 their distributions. While the aforementioned datasets either contain one giant graph for node classification and link prediction, or a set of graphs from different distributions for graph classification.

Graph generation and transformation have been increasingly drawing attentions from the community due to its significant roles in various domains. Though many advanced methods have been proposed, there are only limited number of datasets for this research topics. Enzyme dataset [28], ProFold dataset [29] and Protein dataset [30] are used for protein structure generation. ZINC molecule database is borrowed to generate optimal molecules that have desired properties [20]. Moreover, a few synthetic datasets are also generated for graph generation tasks to learn graph distributions, such as Erdos-Renyi graphs [31] and Waxman random graphs [29]. There exist few data collections that systematic organize the graph generation datasets from different domains.

3 Graph Generation and Transformation

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 $E \in \mathbb{R}^{N \times N \times K}$ that assigns attributes to each edge. D and K are dimensions of node attributes and edge attributes, respectively.

3.1 Graph Generation

Graph generation aims to sample novel graphs via well-designed probabilistic models [7]. 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.

Recent studies on graph generation could be divided into two branches, (1) one-shot generation, (2) sequential generation, based on the their choices of the generation process. Specifically, one-shot generation builds probabilistic matrices for the generated graph features which the graph structures could be obtained by taking the maximum probability nodes and edges in one shot [18, 32, 19, 33]. While sequential generation, formulates graph generation as a sequential process and generates nodes and edges one by one [34, 35, 36, 37].

3.2 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 $(\mathcal{V}', \mathcal{E}', E', F')$ represents a graph in target domain.

Based on the entities transformed in the transformation process, problems regarding graph transformation can be divided into three main scenarios: (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 (3) *node-edge co-transformation*, both the node and edge information can change during the transformation process.

Recent works cover each of three categories of graph transformation models. Interaction networks is a node-transformation technique that provides reasoning on objects, relations and physics [38]. DCRNN integrates diffusion convolution with a seq2seq framework to handle node transformation [39]. Graph Convolutional Policy Network is proposed for modeling chemical reactions. DCGAN has been used for generating novel protein structure [40]. GC-GAN can handle malware cyber-network synthesis [41]. For the node-edge co-transformation, JT-VAE [20] and Mol-CycleGAN [42] are designed for molecule optimization. DG-DAGRNN is employed to generalize stacked RNNs on sequences on directed acyclic graph structures [43].

4 Descriptions of GraphGT Benchmark Datasets

4.1 Taxonomy

Our GraphGT Benchmark covers 36 datasets from various domains and tasks. The taxonomy with respect to different domains is shown in Figure 2, 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 (AI), chemistry, engineering and social science. Moreover, the taxonomy by different tasks is illustrated in Figure 3. 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.

4.2 Dataset Details

In this section, we provide the specifications of representative datasets spanning different subjects introduced in Figure 2. Their potential use in tasks such as graph generation or transformation tasks will also be provided. 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 C.

4.2.1 Biology

Motivation. In biology domain, we have two subjects which are proteins and brain networks. Proteins are essential to all lives, and are highly related to significant biomedicine-related tasks, such as protein design [57] and drug design [58, 59, 60]. De novo protein design [61] is a promising field the explores the full sequence space which is estimated 20^{200} possible amino-acid sequences for only a 200-residue protein with the guidance of physical principles of protein folding. In addition to protein structure, brain networks include two major types of connectivities, structural and functional, which reflect the fiber nerve connectivity and co-activation relations, respectively, among different

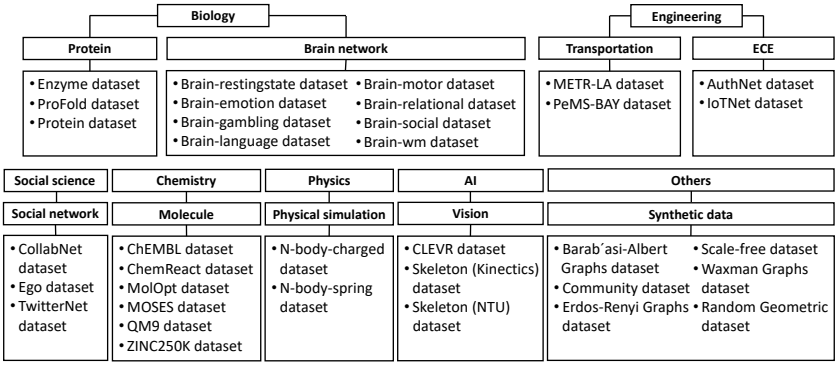

Figure 2: GraphGT Benchmark datasets by domains.

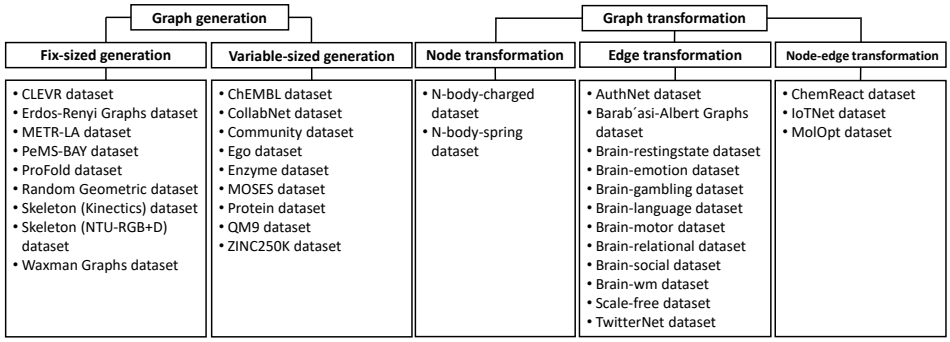

Figure 3: GraphGT benchmark datasets by tasks.

regions of human brains. Understanding and modeling brain networks and the correlations between structural connectivity and functional connectivity are crucial tasks in neuroscience [62].

Tasks. Protein structures can be considered as graphs where amino acids as nodes and contacts as edge connections. Generating novel proteins grounds up to tackle challenges in biomedicine and nanotechnology [61, 57, 58, 63, 64, 65, 64]. In a brain network, the brain regions are represented as nodes and the connectivity between each pair of regions are represented as edges. The graph transformation model can assist understanding the transformation from structural connectivity to resting-state or task-specific functional connectivities in the human brain [31].

Dataset Construction. We reformat 3 protein structure datasets for graph generation and 8 brain network datasets for graph transformation in GraphGT. For protein data, we start from the amino acid coordinates, and then extract graphs of protein structures according to mutual distances of amino acids. The node feature (type of amino acids) are also extracted and recorded in GraphGT. We construct 7 brain network datasets by performing standard neuroimage processing, time series processing, and network construction on both types of connectivities from the magnetic resonance imaging (MRI) data to obtain brain graphs, with edge attributes as Pearson correlation between two regions and node attributes as node index. We also reformat one brain network dataset (Brain-restingstate) that has already been used for graph transformation task [31].

4.2.2 Physics

Motivation. Physical simulation is a significant technique to explore interactions among objects with natural forces. Specific physical systems, such as dynamical systems [49], can be formed into graph structures. The dynamics of a physical system can be seen as a group of interaction components, in which complex dynamics occur at both individual level and in the system as a whole [49]. One could utilize the graph transformation methods to observe the evolution of a physical system.

Tasks. The dynamics of a physical system can be regarded as a graph, in which nodes represent components and edges represent their interactions. Graph transformation models have been applied to physical systems to generate possible conditions of the system sequentially [49, 66, 67]. Work in [68] utilize deep generative models to simulate physically realistic realizations of the cosmic web. Work in [69] introduces generative models in N-body simulations that pushes closer the ideas of deep generative models to practical use in cosmology.

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 [44]	Molecules	133,885	∼ 9	∼ 19	Y	N	Y	N	Y	3D	N	Y
ZINC250K [45]	Molecules	249,455	∼ 23	∼ 50	Y	N	Y	N	Y	3D	N	Y
MOSES [46]	Molecules	193,696	∼ 22	∼ 47	Y	N	Y	N	Y	3D	N	Y
MolOpt [47]	Molecules	229,473	∼ 24	∼ 53	Y	N	Y	N	Y	3D	N	Y
ChEMBL [48]	Molecules	1,799,433	∼ 27	∼ 58	Y	N	Y	N	Y	3D	N	Y
ChemReact [31]	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 [28]	Proteins	600	∼33	∼62	Y	N	N	N	Y	N	N	Y
ProFold [29]	Proteins	76,000	8	∼40	Y	N	N	N	Y	3D	Y	Y
Brain-restingstate [31]	Brain networks	823	68	2274	N	N	Y	Y	Y	N	N	Y
Brain-emotion [31]	Brain networks	811	68	2278	N	N	Y	Y	Y	N	N	Y
Brain-gambling [31]	Brain networks	818	68	2278	N	N	Y	Y	Y	N	N	Y
Brain-language [31]	Brain networks	816	68	2278	N	N	Y	Y	Y	N	N	Y
Brain-motor [31]	Brain networks	816	68	2278	N	N	Y	Y	Y	N	N	Y
Brain-relational [31]	Brain networks	808	68	2278	N	N	Y	Y	Y	N	N	Y
Brain-social [31]	Brain networks	816	68	2278	N	N	Y	Y	Y	N	N	Y
Brain-wm [31]	Brain networks	812	68	2278	N	N	Y	Y	Y	N	N	Y
N-body-charged [49]	Physical simulation networks	3,430,000	25	∼3	Y	N	N	N	Y	2D	Y	Y
N-body-spring [49]	Physical simulation networks	3,430,000	5	∼10	Y	N	N	N	Y	2D	Y	Y
CLEVR [50]	Scene graphs	85,000	6	∼40	Y	Y	Y	N	Y	3D	N	N
Skeleton (Kinectics) [51]	Skeleton graphs	260,000	18	17	N	N	N	N	Y	2D	Y	Y
Skeleton (NTU-RGB+D) [52]	Skeleton graphs	56,000	25	24	N	N	N	N	Y	3D	Y	Y
METR-LA [53]	Traffic networks	34,272	325	2,369	Y	Y	Y	N	Y	GCS	Y	Y
PeMS-BAY [54]	Traffic networks	50,112	207	1,515	Y	Y	Y	N	Y	GCS	Y	Y
AuthNet [41]	Authen. networks	114/412	50/300	∼3/∼7	N	Y	Y	N	Y	N	N	Y
IoTNet [31]	IoT networks	343	20/40/60	∼220/∼630/∼800	Y	N	Y	N	Y	N	N	Y
CollabNet [55]	Collab. networks	2,361	303,308	207,632	N	N	N	N	Y	GCS	Y	Y
Ego [34]	social networks	757	∼145	∼335	N	N	N	N	Y	N	N	N
TwitterNet [56]	social networks	2,580	300	0.5	N	N	N	N	Y	N	N	N
Barab'asi-Albert Graphs [31]	Synthetic networks	1,000	20/40/60	∼60/∼190/∼300	Y	N	N	N	Y	N	N	N
Erdos-Renyi Graphs [31]	Synthetic networks	1,000	20/40/60	∼100/∼200/∼400	Y	N	N	N	Y	N	N	N
Scale-Free [41]	Synthetic networks	10,000	10/20/50/100/150	20/ 40/ 100/ 200/ 320	N	Y	N	N	Y	N	N	N
Community [34]	synthetic networks	3,000	64	∼340	N	N	N	N	Y	N	N	N
Random Geometric [29]	Synthetic networks	9,600	25	∼350	Y	N	N	N	Y	Y	Y	Y
Waxman Graphs [29]	Synthetic networks	9,600	25	∼250	Y	N	N	N	Y	Y	Y	Y

Dataset Construction. We re-purpose two datasets that have never been tried on graph transformation tasks prior to our efforts. We start from velocities and coordinates of each particle and merge them into a single structure with node velocities as node features. Moreover, we extract temporal features from the temporal array contained in original datasets.

4.2.3 Artificial Intelligence

Motivation. Graph-structured data are widely employed in computer vision, a sub-field of AI. We store two most common graph-structured data from computer vision in GraphGT which are skeleton graphs and scene graphs. For example, generating scene graphs is of great importance to understand the relationship in a scene (i.e. image) [70]. In addition to scene graph generation, generating new human skeleton graphs also has a wide range of applications in computer vision, graphics and games, where characters could be generated and interact with human players [71, 72].

Tasks. In a scene graph, objects are represented as nodes and the relationship between pairs of objects is represented as edges. Graph generation models can be applied to the scene graph to help the community understand the relationship between objects in a scene, e.g. generating scene graphs with different relationships (man riding a horse vs. man standing by a horse). In a human skeleton graph, joints are represented as nodes and skeletons between each pair of joints are represented as edges. Similarly, graph generation models can be designed for skeleton graph to help the community approach interactions between human players and characters in a video (e.g. generating AI players with realistic gestures and movements).

Dataset Construction. We re-purpose one dataset for the scene graph and two datasets for skeleton graph that have not been used for graph generation tasks. For the scene graph, we start from the CLEVR dataset containing 10 objects in the image with different 3D locations. Then we form labeled

directed graphs with different shape of objects as the node feature and relative location between two objects as the edge feature. For skeleton graphs, we start from video clips of human action datasets, and then use OpenPose toolbox to generate skeleton with location and joints for each subject. The temporal information is also recorded and wrapped into our data as the temporal feature.

4.2.4 Chemistry

Motivation. Chemistry is another subject in which graph generation and transformation play critical roles for generating optimal molecules or predicting products of chemical reactions [20, 31, 73, 74]. The chemical space, drug-like molecules are vast and estimated to 10^{60} [75]. Generating novel molecules with desired properties has great potentials in discovering new drugs and materials. Modeling chemical reactions is another fundamental problem in chemistry which can advance our understanding of the properties of molecules [73].

Tasks. In a molecular graph, atoms are represented as nodes, and bonds are represented as edges. Molecular graph generation has numerous applications in drug discovery and [76] material science [77] to generate optimal molecules. Moreover, learning the transformation from the reactants to the products can help the community better understand the mechanism of chemical reactions [73].

Dataset Construction. We reformat 6 datasets in chemistry by converting SMILES sequence into molecular structures. Then the molecular structures are converted into graphs with atoms as nodes and chemical bonds as edges. Atom and bond type serve as node and edge feature respectively.

4.2.5 Engineering

Motivation For the engineering field, we provide datasets corresponding to two domains, transportation system and electrical and computer engineering (ECE). First of all, a few graph representation learning methods such as graph neural networks have been applied to transportation research such as traffic prediction [78, 39]. In addition to graph representation learning tasks, graph generative models in machine learning have started experiencing increase in recent years, for tasks like human mobility generative modeling [79] given that a number of tasks can be formalized into a graph generation or transformation problem in the field of engineering. The road system can also be considered as graphs where road segments and interactions are connected, for which the graph generative models can be employed for generating newly designed networks [80].

Tasks. In internet network, graphs contain nodes representing devices, and edges representing connection between two devices. The malware confinement over the internet can be treated as a graph transformation problem to generate optimal status of network that limits malware propagation [31]. Traffic networks contain graphs with nodes as speed sensors and edges as roads. Traffic networks can be employed with graph generation models for designing new and efficient traffic networks.

Dataset Construction. We reformat the malware dataset by adopting the initial attacked networks (i.e., the Internet of Things) as the input graphs, with nodes representing devices and edges representing their connections. Malware confinement status are extracted as node features and distances between two devices are edge features. We also split the dataset according to their graph sizes for different graph transformation purposes. We reformat two transportation datasets by extracting them from LA-Metro and PeMS projects, respectively. We extract sensors as graph nodes an roads as edges, with traffic speed as the node feature. We also extract GCS spatial features and temporal features in the dataset.

4.2.6 Social Science

Motivation. Social networks are an important type of graphs where people or other subjects are connected by relationships such as friendship and co-authorship, and have been widely explored in social science, statistics, and physics with network (generative) modeling techniques. The advancement of graph generative models further stimulate the social network research by handling different aspects of the data. For example, DYMOND achieves graph generation on social networks by borrowing building blocks of network structure to capture long-range interactions [81]. Another graph generative model, TagGen, can preserve both structural and temporal information in the process of modeling interactions in the social network [82].

Tasks. Social networks can be formalized into graphs in which social subjects are nodes and their relationships are edges. The community network has been used to on graph generative models so that the relationship between people or community could be modeled and understood [34].

Dataset Construction. We reformat Ego dataset from Citeseer dataset. Nodes represent documents and edges represent citation relationships. We also re-purpose TwitterNet from [56]. Both datasets do not have node or edge attributes. We construct from scratch the graphs of CollabNet by selecting

authors as nodes and co-authorships as edges. To cut the graphs into pieces, we generate sub-graphs based on the fields of study of papers. For each field, we generate one spatio-temporal graphs.

4.2.7 Synthetic

Motivation. The limited amount of available data in the real world, especially graph data for specific geometric properties [83, 84, 85] for graph generation and transformation problems, limits the advance of the field. Synthetic data is a way to overcome this obstacle and prolong the march of progress in graph generation and transformation tasks. This motivate us to reformat a few simulated synthetic datasets in GraphGT to accommodate various needs of the community for evaluating graph generation and transformation tasks.

Tasks. Synthetic datasets contain graphs corresponding to various geometric properties, including scale-free graphs, Erdos-Renyi graphs, random geometric graphs and so on. A huge amount of works regarding graph generation and transformation have been using synthetic datasets to evaluate their models. NEC-DGT is evaluated with Barab'asi-Albert graphs and Erdos-Renyi graphs [31]. Another graph transformation model, GT-GAN, is evaluated by scale-free graphs [41].

Dataset Construction. We reformat synthetic datasets by converting the original sparse matrices into dense matrices, and reshaping them into predefined dimensions. All synthetic datasets are simulated based on specific geometric properties or laws.

5 Benchmark Experiments

5.1 Graph Generation

5.1.1 Evaluation Metrics

The evaluation of graph generation performance has been widely recognized as a challenging tasks [34, 37] and there lacks a unified framework that can provide comprehensive evaluation procedures and metrics. Following the survey of graph generation [7], we enhanced our deployed API with easy-to-use evaluation tools. The evaluation metrics in GraphGT is elaborated as follows.

In **statistics-based** evaluation metrics, the quality of the generated graphs is accessed 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.

In **self-quality based** evaluation, the quality of the generated graphs, validity, uniqueness and novelty, are measured. The definition and calculation of the three metrics are provided as follows: (1) *Validity*: validity evaluates graphs by judging whether they preserve specific properties. For example, for cycles graphs/tree graphs, the validity is calculated as the percentage of generated graphs that are cycles or trees [8]. For molecule graphs, validity is the percentage of chemically valid molecules based on domain-specific rules [36]. (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 [86, 8, 36]. 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 [86]. 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.

Table 2: Quantitative evaluation and comparison on spatial network generation tasks by different deep generative models on graphs ("Deg." is short for degree distribution. "Clus." is short for clustering coefficient distribution. "Orbit." is short for average orbit counts statistics.).

Method →	GraphRNN			GraphVAE			GraphGMG		
Dataset ↓	Deg. (%)	Clus. (%)	Orbit. (%)	Deg. (%)	Clus. (%)	Orbit. (%)	Deg. (%)	Clus. (%)	Orbit. (%)
Waxman	1.20	1.74	0.87	120.14	144.22	109.72	26.44	41.58	21.15
Random Geometric	1.09	19.19	2.80	88.27	95.52	102.71	57.12	111.94	71.32
CLEVR	56.89	2.66	61.19	0.00	0.00	0.00	126.96	163.53	180.65
METR-LA	193.11	196.69	165.86	-	-	-	-	-	-
PeMS-BAY	172.97	173.37	159.68	-	-	-	-	-	-
ProFold	1.10	0.38	0.09	114.60	109.02	84.78	5.55	44.61	4.55
Skeleton (Kinetics)	$< 10^{-5}$	0.00	$< 10^{-5}$	200.00	200.00	200.00	9.84	0.00	0.06
Skeleton (NTU-RGB+D)	$< 10^{-5}$	0.00	$< 10^{-5}$	200.00	200.00	200.00	120.31	0.27	2.31
CollabNet	-	-	-	-	-	-	-	-	-
N-body-charged	172.93	0.00	0.00	0.00	0.00	0.00	37.83	75.48	2.76
N-body-spring	3.17	1.86	0.02	141.06	123.22	5.71	127.42	49.46	0.75
Ego	66.44	129.82	64.18	-	-	-	-	-	-
Community	19.61	55.46	57.09	-	-	-	-	-	-
Protein	2.57	5.27	1.27	-	-	-	-	-	-
Enzyme	0.81	1.64	0.88	-	-	-	-	-	-

5.1.2 Benchmark Results

For graph generation, we benchmark 16 graph generation datasets in GraphGT with GraphRNN [34], GraphVAE [18], and GraphGMG [8], three common graph generation baselines. The detailed descriptions of each baseline models can be found in Appendix D. We evaluate the performance of the graph generative models on three statistics-based metrics, degree distribution, clustering coefficient distribution and orbit counts statistics. For efficiency problem, GraphVAE and GraphGMG cannot scale to multiple large datasets, e.g. METR-LA, Protein, Enzyme, etc. Note that the CollabNet is too large even for GraphRNN to scale. From Table 2, we can observe that GraphRNN outperforms GraphVAE and GraphGMG in most of the datasets. Notably, GraphRNN takes the advantage of sequential graph generation which allows scaling to large graphs, while GraphVAE cannot due to its costly one-shot generation method. Additioanlly, GraphRNN works extraordinarily well on relatively small graphs datasets, e.g. Profold, N-body, Skeleton, while performs worse on large graphs like traffic networks. GraphVAE performs very well in two particular datasets which are CLEVR and N-body-charged which both of them are very small and the simulation processes are stochastic. GraphGMG performs well in specifically one skeleton graph and one protein dataset which both of the graph structures are fixed and simple. Additionally, GraphVAE outperforms the sequence-based models on CLEVR and N-body-charged datasets. We believe that it is easier for an one-shot generation method to learn topology which is related to spatial locations since it doesn't have to learn a sequence-dependent process.

5.2 Graph Transformation

5.2.1 Evaluation Metrics

In **Graph-property-based** evaluation, we directly compare each generated graph to its target graph via the following metrics: (1) random-walk kernel similarity by using the random-walk based graph kernel [87]; (2) combination of Hamming and Ipsen-Mikhailov distances(HIM) [88]; (3) spectral entropies of the density matrices; (4) eigenvector centrality distance [89]; (5) closeness centrality distance [90]; (6) Weisfeiler Lehman kernel similarity [91]; (7) Neighborhood Sub-graph Pairwise Distance Kernel [92] by matching pairs of subgraphs with different radii and distances; (8) Jensen–Shannon distance, (9) Bhattacharyya distance and (10) Wasserstein distance by measuring distance of node degrees of two graphs.

In **Mapping-relationship-based** evaluation, we measure 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 [7]: (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 [93]. (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 [41]. 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

Table 3: Quantitative evaluation and comparison on transformation tasks by different deep transformation models on graphs ("JS-dist." is the Jensen–Shannon distance. "BH-dist." is the Bhattacharyya distance. "WS-dist." is the Wasserstein distance.).

Method →	**Interaction Network**			**NEC**-DGT		
Dataset ↓	**JS-dist. (%)**	**BH-dist. (%)**	**WS-dist. (%)**	**JS-dist. (%)**	**BH-dist. (%)**	**WS-dist. (%)**
AuthNet	1.04	0.01	0.33	82.81	95.88	24.59
Barab'asi-Albert Graphs	4.50	0.21	5.12	66.87	59.39	36.84
Brain-restingstate	11.17	1.26	13.26	11.39	1.31	18.24
Brain-emotion	12.63	1.61	15.78	12.83	1.66	12.58
Brain-grambling	12.55	1.59	15.73	12.82	1.66	26.54
Brain-language	12.23	1.51	15.24	12.56	1.60	16.51
Brain-motor	11.88	1.43	14.69	12.14	1.49	31.04
Brain-relational	12.26	1.52	15.23	12.50	1.58	35.62
Brain-social	12.09	1.48	14.97	12.34	1.54	141.58
Brain-wm	12.23	1.51	15.24	12.48	1.58	37.31
Scale-free	1.19	0.01	0.42	79.13	83.00	21.71
TwitterNet	0.01	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$	6155.10
N-body-charged	0.12	$< 10^{-3}$	0.14	4.37	0.21	47.52
N-body-spring	0.05	$< 10^{-3}$	0.07	4.50	0.20	53.20
ChemReact	0.94	$< 10^{-3}$	0.27	77.84	79.92	0.6714
IoTNet	17.01	3.01	19.32	65.39	55.90	2572.62
MolOpt	0.71	0.01	0.11	82.67	94.89	19.97

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.

5.2.2 Benchmark Results.

Here, 17 transformation datasets are benchmarked for graph transformation tasks in GraphGT. Two state-of-the-art graph transformation models, Interaction network (IN) [38] and Node-Edge Co-evolving Deep Graph Translator (NEC-DGT) [31] are borrowed to analyze these datasets. Three metrics, Jensen–Shannon distance, Bhattacharyya distance and Wasserstein distance, are used to measure the distance between the distribution of generated graphs and target graphs. Details regarding the experimental settings can be found in Appendix D. We find that two models have a close performance regarding graph transformation on most datasets. This is not surprising since two models follow similar philosophies to handle node interactions in the graph. With the Interaction Network, the smallest Jensen–Shannon and Bhattacharyya distance are achieved on TwitterNet, which is aligned with NEC-DGT. TwitterNet also has the closest Wasserstein distance, whether Brain-emotion has the closest Wasserstein distance for NEC-DGT. This difference might originate from the capacity to handle node or edge features of two models, or different hyper-parameter settings. Interaction Networks can handle edge attributes, which are available for Brain-emotion dataset but not for TwitterNet dataset, whereas NEC-DGT can handle both node and edge attributes, neither of which are available for TwitterNet. We also find that, for the same model, datasets from different domains have different performances. We observe a relatively large distances regarding three metrics for 8 brain network datasets compared with most other datasets when being evaluated by Interaction Network. However, these 8 datasets have a relatively smaller distance when being evaluated by NEC-DGT. This reflects the complexity of the brain network domain [94] that needs more advanced models to be handled, such as NEC-DGT. N-body-charged and N-body-spring datasets have a generally smaller distances compared with most other datasets when being evaluated by both models. This results from the relatively small graph size in physical simulation domain (Table 3).

6 Conclusion

We introduce GraphGT, a large dataset collection for graph generation and transformation problems. GraphGT covers datasets in 9 domains across 6 subjects, in which CollabNet dataset and 7 brain network datasets are collected and constructed from scratch for graph generation and transformation. Another 8 datasets are re-purposed by us from other applications into graph generation and transformation tasks for the first time. The remaining are from very different domains that share quite different terminology, formats, and data structures, which are reformatted by us to a unified format for the first time for easy access and use in a standardized manner. In addition, we provide 3 types of Python APIs, including dataset downloader, graph generation data processor, graph transformation data processor and evaluator, for users to query and access datasets according to specific disciplines, domains and applications per their interests. Finally, we provide 16 graph generation benchmark results and 17 graph transformation benchmark results 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] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph Contrastive Learning with Adaptive Augmentation. In *Proceedings of The Web Conference 2021*, WWW '21, pages 2069–2080, New York, NY, USA, April 2021. Association for Computing Machinery.

[6] Dongkuan Xu, Wei Cheng, Dongsheng Luo, Haifeng Chen, and Xiang Zhang. Infogcl: Information-aware graph contrastive learning. In *NeurIPS*, 2021.

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

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

[9] 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.

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

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

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

[13] 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.

[14] 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.

[15] Dongkuan Xu, Junjie Liang, Wei Cheng, Hua Wei, Haifeng Chen, and Xiang Zhang. Transformer-style relational reasoning with dynamic memory updating for temporal network modeling. In *AAAI*, volume 35, pages 4546–4554, 2021.

[16] 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.

[17] 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.

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

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

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

[21] 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.

[22] 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.

[23] 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.

[24] Faezeh Faez, Yassaman Ommi, Mahdieh Soleymani Baghshah, and Hamid R Rabiee. Deep graph generators: A survey. *IEEE Access*, 9:106675–106702, 2021.

[25] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

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

[27] 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.

[28] 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.

[29] 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.

[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] 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.

[32] Chengxi Zang and Fei Wang. Moflow: an invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 617–626, 2020.

[33] Meng Liu, Keqiang Yan, Bora Oztekin, and Shuiwang Ji. Graphebm: Molecular graph generation with energy-based models. *arXiv preprint arXiv:2102.00546*, 2021.

[34] 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.

[35] Chence Shi, Minkai Xu, and Zhaocheng Zhu et al. Graphaf: a flow-based autoregressive model for molecular graph generation. 2020.

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

[37] Youzhi Luo, Keqiang Yan, and Shuiwang Ji. Graphdf: A discrete flow model for molecular graph generation. *arXiv preprint arXiv:2102.01189*, 2021.

[38] Peter W Battaglia, Razvan Pascanu, Matthew Lai, Danilo Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. *arXiv preprint arXiv:1612.00222*, 2016.

[39] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

[40] Namrata Anand and Possu Huang. Generative modeling for protein structures. 2018.

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

[42] Łukasz Maziarka, Agnieszka Pocha, Jan Kaczmarczyk, Krzysztof Rataj, Tomasz Danel, and Michał Warchoł. Mol-cyclegan: a generative model for molecular optimization. *Journal of Cheminformatics*, 12(1):1–18, 2020.

[43] Saeed Amizadeh, Sergiy Matusevych, and Markus Weimer. Learning to solve circuit-sat: An unsupervised differentiable approach. In *International Conference on Learning Representations*, 2018.

[44] 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.

[45] 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.

[46] 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.

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

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

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

[55] 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.

[56] 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.

[57] John Ingraham, Vikas K Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. 2019.

[58] Paweł Śledź and Amedeo Caflisch. Protein structure-based drug design: from docking to molecular dynamics. *Current opinion in structural biology*, 48:93–102, 2018.

[59] Yuanqi Du, Xiaojie Guo, Amarda Shehu, and Liang Zhao. Interpretable molecule generation via disentanglement learning. In *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, pages 1–8, 2020.

[60] Xiaojie Guo, Yuanqi Du, and Liang Zhao. Property controllable variational autoencoder via invertible mutual dependence. In *International Conference on Learning Representations*, 2020.

[61] Po-Ssu Huang, Scott E Boyken, and David Baker. The coming of age of de novo protein design. *Nature*, 537(7620):320–327, 2016.

[62] Danielle S Bassett and Olaf Sporns. Network neuroscience. *Nature neuroscience*, 20(3):353–364, 2017.

[63] Alexey Strokach, David Becerra, Carles Corbi-Verge, Albert Perez-Riba, and Philip M Kim. Fast and flexible protein design using deep graph neural networks. *Cell Systems*, 11(4):402–411, 2020.

[64] Xiaojie Guo, Sivani Tadepalli, Liang Zhao, and Amarda Shehu. Generating tertiary protein structures via an interpretative variational autoencoder. *arXiv preprint arXiv:2004.07119*, 2020.

[65] Taseef Rahman, Yuanqi Du, Liang Zhao, and Amarda Shehu. Generative adversarial learning of protein tertiary structures. *Molecules*, 26(5):1209, 2021.

[66] Masanobu Horie, Naoki Morita, Toshiaki Hishinuma, Yu Ihara, and Naoto Mitsume. Isometric transformation invariant and equivariant graph convolutional networks. *arXiv preprint arXiv:2005.06316*, 2020.

[67] Ali Hariri, Darya Dyachkova, Sergei Gleyzer, Mariette Awad, Daria Morozova, and Pangea Formazione. Graph generative models for fast detector simulations in particle physics. In *Machine Learning and the Physical Sciences Workshop atNeurIPS*, volume 1, pages 1–6, 2020.

[68] Andres C Rodríguez, Tomasz Kacprzak, Aurelien Lucchi, Adam Amara, Raphael Sgier, Janis Fluri, Thomas Hofmann, and Alexandre Réfrégier. Fast cosmic web simulations with generative adversarial networks. *Computational Astrophysics and Cosmology*, 5(1):1–11, 2018.

[69] Nathanaël Perraudin, Ankit Srivastava, Aurelien Lucchi, Tomasz Kacprzak, Thomas Hofmann, and Alexandre Réfrégier. Cosmological n-body simulations: a challenge for scalable generative models. *Computational Astrophysics and Cosmology*, 6(1):1–17, 2019.

[70] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph r-cnn for scene graph generation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 670–685, 2018.

[71] Sijie Yan, Zhizhong Li, Yuanjun Xiong, Huahan Yan, and Dahua Lin. Convolutional sequence generation for skeleton-based action synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4394–4402, 2019.

[72] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5933–5942, 2019.

[73] Kien Do, Truyen Tran, and Svetha Venkatesh. Graph transformation policy network for chemical reaction prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 750–760, 2019.

[74] Dai Hai Nguyen and Koji Tsuda. A generative model for molecule generation based on chemical reaction trees. *arXiv preprint arXiv:2106.03394*, 2021.

[75] Lars Ruddigkeit, Ruud Van Deursen, Lorenz C Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875, 2012.

[76] Gisbert Schneider. Automating drug discovery. *Nature reviews drug discovery*, 17(2):97–113, 2018.

[77] Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365, 2018.

[78] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*, 2017.

[79] Liming Zhang, Liang Zhao, Shan Qin, Dieter Pfoser, and Chen Ling. Tg-gan: Continuous-time temporal graph deep generative models with time-validity constraints. In *Proceedings of the Web Conference 2021*, WWW '21, page 2104–2116, New York, NY, USA, 2021. Association for Computing Machinery.

[80] James Jian Qiao Yu and Jiatao Gu. Real-time traffic speed estimation with graph convolutional generative autoencoder. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3940–3951, 2019.

[81] Giselle Zeno, Timothy La Fond, and Jennifer Neville. Dymond: Dynamic motif-nodes network generative model. In *Proceedings of the Web Conference 2021*, pages 718–729, 2021.

[82] Dawei Zhou, Lecheng Zheng, Jiawei Han, and Jingrui He. A data-driven graph generative model for temporal interaction networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 401–411, 2020.

[83] 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.

[84] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[85] Sergey I Nikolenko et al. Synthetic data for deep learning. *arXiv preprint arXiv:1909.11512*, 3, 2019.

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

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

[88] 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.

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

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

[91] 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.

[92] 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.

[93] 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.

[94] Alex Fornito, Andrew Zalesky, and Edward Bullmore. *Fundamentals of brain network analysis*. Academic Press, 2016.

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

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

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

[98] 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.

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

[100] 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.

[101] 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.

[102] 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.

[103] 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.

[104] 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.

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

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

[107] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *International Conference on Machine Learning*, pages 1945–1954. PMLR, 2017.

[108] Qi Liu, Miltiadis Allamanis, and Marc Brockschmidt et al. Constrained graph variational autoencoders for molecule design. In *NeurIPS'2018*, pages 7795–7804, 2018.

[109] Shion Honda, Hirotaka Akita, and Katsuhiko Ishiguro et al. Graph residual flow for molecular graph generation. *arXiv preprint arXiv:1909.13521*, 2019.

[110] Tengfei Ma, Jie Chen, and Cao Xiao. Constrained generation of semantically valid graphs via regularizing variational autoencoders. In *NeurIPS'2018*, pages 7113–7124, 2018.

[111] Daniel Flam-Shepherd, Tony Wu, and Alan Aspuru-Guzik. Graph deconvolutional generation. *arXiv preprint arXiv:2002.07087*, 2020.

[112] Dongmian Zou and Gilad Lerman. Encoding robust representation for graph generation. In *IJCNN'2019*, pages 1–9, 2019.

[113] Aditya Grover, Aaron Zweig, and Stefano Ermon. Graphite: Iterative generative modeling of graphs. In *International conference on machine learning*, pages 2434–2444. PMLR, 2019.

A Key Information about GraphGT

A.1 Dataset Documentation

We provide detailed documentation of dataset collection, processing, task for each dataset both in section C 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 Format

We store each of the dataset in a Numpy[3] array format. For different datasets with different information available as shown in Table 1. For all the datasets, each has at most five types of features available including adjacency matrices, node features, edge features, spatial features, and labels. Among all the features, *adjacency matrices* denote the edge connections between pairs of nodes, *node features* denote features attaching to each node, *edge features* denote features attaching to each edge connection, *spatial features* denote the spatial geometry of a graph (in most of the cases, they are coordinates attaching to each node), *labels* denote either node-level or graph-level labels of a graph. For temporal graphs, we store two versions of the graphs, which one flattens and shuffles all the snapshots of the temporal graphs, and the other one keeps the temporal dimension and order. For graph transformation datasets, we store both the source and the target graph and available features separately.

C Dataset Details

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

[3] https://numpy.org/doc/

C.1 Molecules

We have 6 molecule datasets, in which 4 (QM9 [44], ZINC250K [45], MOSES [46], ChEMBL [48]) for graph generation and 2 (MolOpt [47], ChemReact [31]) 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 [44] dataset is an enumeration of around 134,000 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. In QM9 dataset, each graph contains approximately 9 nodes and 19 edges. A node in QM9 represents an atom with atom type as the node feature. An edge in QM9 dataset represents a bond in the molecule with bond type as the edge feature. Moreover, QM9 dataset contains the 3D spatial feature for each graph. In GraphGT, the QM9 dataset has been reformatted as adj.npy, edge_feat.npy, label.npy, node_feat.npy and spatial.npy that contain molecular structure information, node features, edge features and spatial features.

The information of QM9 is initially stored in .xyz files separately for each molecule. We use Python to process the SMILE of each molecule and convert the molecule graph to Numpy formats.

ZINC250K [45] 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. Each graph in ZINC250K dataset contains approximated 23 nodes and 50 edges. In ZINC250K dataset, each node represents an atom, with atom type as the node feature. An edge in 250K dataset represents a bond in the molecule with bond type as the edge feature. 250K dataset also contains 3D spatial feature for each graph. In GraphGT, the ZINC250K dataset has been reformatted as adj.npy, edge_feat.npy, label.npy, node_feat.npy and spatial.npy that contain molecular structure information, node features, edge features and spatial features.

ZINC dataset is stored in one .csv file including 249,455 molecules. After reading the data by Python, we process the SMILE of each molecule to convert the data to a graph. And all the graphs are saved in .npy format.

Molecular Sets (MOSES) [46] 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, and contains 193,696 molecules in total. Each graph in the dataset contains around 22 nodes and 47 edges. In MOSES dataset, each node represents an atom, with atom type as the node feature. An edge in MOSES dataset represents the bond in the molecule with bond type as the edge feature. MOSES datasets also contains 3D spatial features.

The data is originally stored in a .txt file. We first read the data and then process the SMILE of the molecule based on the Python rdkit library. The final data format is saved as .npy files.

ChEMBL [48] 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. ChEMBL contains 1,799,433 graphs in total. Each graph in the dataset contains around 27 nodes and 58 edges. In ChEMBL dataset, each node represents an atom, with atom type as the node feature. An edge in ChEMBL dataset represents the bond in the molecule with bond type as the edge feature. This datasets also contains 3D spatial features.

ChEMBL is originally stored in a .txt file containing all the molecules. We first read the data and then process the SMILE of the molecule based on the Python rdkit library. The final data format is saved as .npy files.

MolOpt [47] dataset extracts translation pairs from the ZINC database in terms of three molecular properties, Penalized logP, Drug-likeness, and Dopamine Receptor. MolOpt contains 229,473 pairs of graphs in total. Each graph in the dataset contains around 24 nodes and 53 edges. In MolOpt dataset, each node represents an atom, with atom type as the node feature. An edge in ChEMBL dataset represents the bond in the molecule with bond type as the edge feature. This datasets also contains 3D spatial features.

This dataset is originally stored in several .csv files and the format of the dataset has been preprocessed. We read the .csv files and convert the SMILE molecules to graphs and then save them as .npy files.

ChemReact [31] dataset has totally 7180 pairs of reactant and product molecule graph in the dataset derived from USPTO dataset. Each graph in the dataset contains around 20 nodes and 16 edges. In ChemReact dataset, each node represents an atom, with atom type as the node feature. An edge in ChemReact dataset represents the bond in the molecule with bond type as the edge feature. This datasets also contains 3D spatial features. [95].

Chemical Reaction dataset is originally stored in several .txt files. The first step for processing the data is to aggregate data from different sources. Then we convert the SMILE of molecules to graph formats, and then save them in .npy files.

C.1.1 License

QM9: CC BY-NC-SA 4.0.

ZINC250K: Free to use for everyone.

MOSES: The dataset is generated by [46], 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.

C.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. Each protein is represented by a graph in Protein dataset, where nodes are amino acids and two nodes are connected if they are less than 6 Angstroms apart. Proteins dataset contains 1,113 graphs in total. Each graph in the dataset contains around 39 nodes and 73 edges. Node feature is contained in the dataset representing the type of amino acids. Protein dataset can be used for attributed graph generation.

Protein dataset is originally stored in several .txt files with the unit of node. We read all .txt files to generate graphs, convert them to Numpy arrays and save them in .npy format.

Enzyme [28] 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. Each graph in the dataset contains around 33 nodes and 62 edges. The node features in the graph represent type of amino acids. This dataset can be employed for attributed graph generation.

Enzyme dataset is originally stored in several .txt files with the unit of node. We read all .txt files to generate graphs, convert them to Numpy arrays and save them in .npy format.

ProFold [29] dataset contains dynamic folding processes of a protein peptide with sequence AGAAAAGA in 38 steps. ProFold contains 76,000 graphs in total. Each graph has 8 nodes and around 40 edges. The node represents amino acid of the protein, and the edge represent the bond between amino acids. 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. This dataset can be used for either attributed graph generation or temporal graph generation.

C.2.1 License

Enzyme: CC-BY-4.0.

ProFold: The dataset is collected by [29]. The license is not specified.

Protein: CC-BY-4.0.

C.3 Brain Networks

The Brain dataset comes from the human connectome project (HCP) [31] 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 [31]. 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 [96] 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. The dataset has been splitted into 8 categories for 8 specific domains, including Brain-restingstate, Brain-emotion, Brain-gambling, Brain-language, Brain-motor, Brain-relational, Brain-social and Brain-wm. All of these datasets can be employed for eight weighted graph transformation or signed graph transformation tasks.

Originally, data is a group of .npz files, containing the structural connectivities for each subject, functional connectivities for each subject, and list of subject IDs for each task using different correlations. Unfortunately, the subjects used are not universal for all tasks, and so we eliminate all but those that appeared in every single task. From there, we simply concatenate all of the functional connectivities from all of the various tasks using FC correlation, and concatenated all of the structural connectivities from all of the various tasks using FC correlation, thus creating FC_concatenated_edge_feat and SC_concatenated_edge_feat. For the adjacency matrix containing .npy arrays, we encounter a small issue; the adjacency matrix is required to be formatted with a specific shape, but that shape is not compatible with the edge feature shape, and so we make the adjacency matrix a placeholder basically. For details please refer to readme.txt.

C.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.

C.4 Physical Simulations

N-body-charged [49] 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-charged dataset contains 3,430,000 graphs in total, each of which contains 25 nodes with around 3 edges. Each node represents a particle and each edge represents interaction between nodes. Node attribute represents node input. 2d spatial features and temporal are included in the dataset. N-body-charged can be used for either attributed graph transformation, spatial graph transformation or temporal graph transformation.

Originally, for the charged dataset, there are separate numpy files for the velocities, edges, and locations of each particle for train, validation, and testing. Then, all velocity arrays(train, valid, test) for the charged dataset were merged into a single one, and the same was done for all of the location arrays, and all of the edge arrays. To convert the charged edge features into adjacency matrices, all nonzero values were turned to ones, and since all particles had some form of connection, that meant all adjacency matrices ended up being all ones for the charged dataset. Then, for each new temporal array we had here, we created a new version: a non-temporal one, where we concatenated the first two dimensions of the array, as the second dimension represented the different temporal instances. For details information, please refer to readme.txt.

N-body-spring [49] 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. N-body-spring dataset contains 3,430,000 graphs in total, each of which contains 5 nodes with around 10 edges. Each node represents a particle and each edge represents interaction between nodes. Node attribute represents node input. 2D spatial features and temporal features are included in the dataset. N-body-spring can be used for either attributed graph transformation, spatial graph transformation or temporal graph transformation.

Originally, for the spring dataset, there were separate numpy files for the velocities, edges, and locations of each particle for train, validation, and testing. There are 5 particles, 5 springs in each graph. Then, all velocity arrays(train, valid, test) for the spring dataset were merged into a single one,

and the same was done for all of the location arrays, and all of the edge arrays. For the springs dataset, we had only ones and zeroes in the edges: connection or no connection, and so we simply took this as our adjacency matrix as well for each matrix in the springs dataset. Then, for each new temporal array we had here, we created a new version: a non-temporal one, where we concatenated the first two dimensions of the array, as the second dimension represented the different temporal instances.

C.4.1 License

N-body-charged: The dataset is simulated by [49], which is under MIT License. The license of the dataset is not specified.

N-body-spring: The dataset is simulated by [49], which is under MIT License. The license of the dataset is not specified.

C.5 Collaboration Networks

CollabNet [55] 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 303,308 nodes and a total of 207,632 of edges. This dataset contains temporal and GCS spatial features, so that the dataset can be used for spatial graph generation and temporal graph generation.

C.5.1 License

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

C.6 Traffic Networks

METR-LA [53] 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. The dataset contains 34,272 graphs, each of which has 325 nodes and 2,369 edges. In METR-LA, each node represent a speed senor and each edge represents a road. The node features of the dataset represent the traffic speed captured by the sensor. The dataset contains GCS spatial features and temporal features. METR-LA can be used for spatial graph generation, temporal graph generation, attributed graph generation and weighted graph generation.

The information of the METR-LA dataset is stored in three files with different formats. We borrow Python to read these data, and convert them to Numpy formats. We then save the data in .npy format.

PeMS-BAY [54] 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. The dataset contains 50,221 graphs, each of which has 207 nodes and 1,515 edges. In PeMS-BAY, each node represent a speed senor and each edge represents a road. The node features of the dataset represent the traffic speed captured by the sensor. The dataset contains GCS spatial features and temporal features. PeMS-BAY can be used for spatial graph generation, temporal graph generation, attributed graph generation and weighted graph generation.

The information of the PeMS-BAY dataset is stored in three files with different formats. We borrow Python to read these data, and convert them to Numpy formats. We then save the data in .npy format.

C.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.

C.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). [97, 41]. There are two subsets of different sizes of graphs (e.g., 50 and 300) in AuthNet dataset with 114 and 412 graphs, respectively. 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). This dataset can be employed for weighted graph generation.

C.7.1 License

AuthNet: The dataset is publically released by LANL [97]. 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.

We collect this dataset from DBLP-Citation-network V12. We chose authors with affiliations, papers with more than one authors, and the time period from 1990 to 2019. To cut the graphs into pieces, we generate sub-graphs based on the fields of study of papers. For each field, we generate one spatio-temporal graph. Then we concatenate and pad all graphs, and save them into Numpy arrays. We save the graphs in .npy format.

C.8 IoT Networks

IoTNet is the malware dataset collected for malware confinement prediction [31]. 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 (i.e., the Internet of Things) as input graphs. Benign activities include MiBench [98] and SPEC2006 [99], 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 [100, 101, 102]. 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. IoTNet can be used for attributed graph generation and weighted graph generation.

The original format of IoTNet contains 1,029 .csv files, we convert them to .npy files, input_adj.npy, input_edge.npy, input_node.npy, target_adj.npy, target_edge.npy, target_node.npy, Iot_20_labels.npy, Iot_40_labels.npy and Iot_60_labels.npy, to contain structure, node features, edge features and labels and to be easily read by Python. The detailed information of the data can be found in the corresponding readme.txt file. To reformat the data, we use glob to read in all .csv files from the directory, and separate the original .csv files into input data and target data; For both input and target data, we get edge feature from the original .csv files, get node feature(0 or 1 for IoTNet) from the diagonals of each file, and get adjacent matrix from the edge feature while setting the diagonals to be 0. For IoTNet, we also split the name and get labels from the name of each .csv file. We then reshaped all arrays into the required dimensions and converted them to NumPy files.

C.8.1 License

IoTNet: The dataset is generated by [31]. The license is not specified.

C.9 Skeleton Graphs

Kinetics [51] 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 [51] 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. This dataset does not contain node or edge attributes, but contain temporal and 2D spatial features to be used in spatial graph generation and temporal graph generation tasks.

The raw Kinetics dataset is stored in a few .json files, and each json file contains information of a single video clip. We traverse all .json files, and concatenate their contents into several Numpy arrays with paddings for short video clips. We then remove extra skeletons, and leave each video clip only one skeleton. Finally, we save the data in .npy array.

NTU-RGB+D [52] 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. NTU-RGB+D does not contain node or edge attributes, but contain temporal and 3D spatial features to be used in 3D spatial graph generation and temporal graph generation tasks.

We process this dataset by the code from github. The dataset is originally stored in a few files, and each contains information of one single video clip. After the same processing process as we do for Kinetics dataset, we save the data in .npy format.

C.9.1 License

Skeleton (Kinectics): CC BY 4.0.

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

C.10 Social Networks

Ego: Ego dataset contains 757 3-hop ego networks extracted from the Citeseer [103]. The number of nodes of the graph in Ego dataset ranges from 50 to 399, and 145 in average. Each graph in Edo has around 335 edges. Nodes represent documents and edges represent citation relationships [34]. Ego does not contain node or edge attributes, and can be used for graph generation tasks.

TwitterNet: The dataset is processed by [56] 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. TwitterNet contains 2,580 graphs in total, each of which has 300 nodes and 0.5 edges in average. This dataset can be employed in graph transformation tasks.

C.10.1 License

Ego: This dataset is extracted from Citeseer [103]. Citeserr is under CC BY-NC-SA 3.0.

TwitterNet: The dataset is obtained from [104]. The license is not specified.

C.11 Scene Graphs

CLEVR [50] dataset provides a dataset for visual question answer, which can be formalized as a spatial-graph dataset. CLEVR dataset contains 85,000 graphs in total. 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. As a result, CLEVR dataset can be employed for attributed graph generation, weighted graph generation and spatial graph generation.

C.11.1 License

CLEVR: CC BY 4.0.

C.12 Synthetic Graphs

Barab'asi-Albert Graphs: This dataset is generated by the Barab'asi-Albert model [31]. 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. Barab'asi-Albert Graphs dataset can be used for attributed graph transformation.

The original Barab'asi-Albert Graphs dataset contains 3,000 .csv files. We reformat them into .npy files, including input_adj.npy, input_edge.npy, input_node.npy, target_adj.npy and target_edge.npy, target_node.npy for the community to use. To reformat the data, we use glob to read in all .csv files from the directory, and separate the original .csv files into input data and target data; For both input and target data, we get edge feature from the original .csv files, get node feature from the diagonals of each file, and get adjacent matrix from the edge feature while setting the diagonals to be 0. We then reshaped all arrays into the required dimensions and converted them to NumPy files.

Community: This dataset is generate by [34] and contains 3,000 two-community graphs, each of which has 64 nodes and around 340 edges. Each community is generated by the Erdos-Renyi model (E-R) [105] 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. This dataset does not have node or edge attributes. Community can be used for graph generation tasks.

Erdos-Renyi Graphs: This dataset is generated by the Erdos-Renyi model with the edge probability of 0.2 [31]. 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. This dataset contains 1,000 graphs in total, and can be used for attributed graph generation.

The original Erdos-Renyi Graphs dataset contains 3,000 .csv files. We reformat them into .npy files, including input_adj.npy, input_edge.npy, input_node.npy, target_adj.npy and target_edge.npy, target_node.npy for the community to use. Detailed information can be found in ER_Readme.trf. To reformat the data, we use glob to read in all .csv files from the directory, and separate the original .csv files into input data and target data; For both input and target data, we get edge feature from the original .csv files, get node feature from the diagonals of each file, and get adjacent matrix from the edge feature while setting the diagonals to be 0. We then reshaped all arrays into the required dimensions and converted them to NumPy files.

Scale-free: This dataset is generated as a directed scale-free network [41], which is a network whose degree distribution follows power-law property [83]. 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. This dataset contains 10,000 graphs in total, and can be splitted into subsets that contains 10, 20, 50, 100, 150 nodes along with 20, 40, 100, 200 and 320 edges, respectively.

The original Scale-free dataset contains 10,000 .csv files and we convert it to .npy files for people to read in Python. The detailed information of the data can be found in the corresponding scale_free_Readne.rtf. To reformat the data, we use glob to read in all .csv files from the directory, and separate the original .csv files into input data and target data; For both input and target data, we get edge feature from the original .csv files, get node feature from the diagonals of each file, and get adjacent matrix from the edge feature while setting the diagonals to be 0. Due to the massive .csv files in the Scale-free Graphs, we optimize to reduce the time complexity in order to process the dataset faster. We then reshaped all arrays into the required dimensions and converted them to NumPy files.

Waxman Graphs: This datase contains graphs generated by the Waxman random graph model that places n nodes uniformly at random in a rectangular domain [106, 29]. 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 [29]. There are 80,000 samples for training and 80,000 for testing. Each graph in the dataset contains 25 nodes and around 250 edges. Waxman Graphs dataset can be used for a few tasks, including attributed graph generation, spatial graph generation and temporal graph generation.

The original Waxman Graphs dataset contains 96,000 graph files saved in Numpy array. We reformat them into .npy files, including adj.npy, edge_feat.npy, label.npy, node_feat.npy, spatial.npy, temporal_adj.npy, temporal_edge.npy, temporal_label.npy, temporal_node.npy and temporal_spatial.npy. The detailed informatiom can be found in waxman_Readme.rtf. To reformat these files, we load the testing and training dataset and converted the sparse matrices to dense matrices. we concatenate the testing and training datasets and reshape them into the required dimensions. To get the version of datasets with temporal dimension, we flattened the NumPy arrays. All datasets were saved as NumPy files eventually.

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 [29]. 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. Each graph in the dataset contains 25 nodes and around 350 edges. Random Geometric Graphs dataset can be used for a few tasks, including attributed graph generation, spatial graph generation and temporal graph generation.

The original Random Geometric Graphs dataset contains 96,000 graph files saved in Numpy array. We reformat them into .npy files, including adj.npy, edge_feat.npy, label.npy, node_feat.npy, spatial.npy, temporal_adj.npy, temporal_edge.npy, temporal_label.npy, temporal_node.npy and temporal_spatial.npy. The detailed informatiom can be found in random_geo_Readme.rtf. To reformat these files, we load the testing and training dataset and converted the sparse matrices to dense matrices. we concatenate the testing and training datasets and reshape them into the required dimensions. To get the version of datasets with temporal dimension, we flattened the NumPy arrays. All datasets were saved as NumPy files eventually.

C.12.1 License

Barab'asi-Albert Graphs: The dataset is generated by [31]. The license is not specified.

Community: The dataset is generated by [34], which is under MIT License. The license of the dataset is not specified.

Erdos-Renyi graphs: The dataset is generated by [31]. The license is not specified.

Scale-free: The dataset is generated by [41]. The license is not specified.

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

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

D Benchmark Results

We benchmark all the datasets with graph generation and transformation models. For graph generation task, we conduct experiments on three models, GraphRNN [34], GraphVAE [18], GraphGMG [8]. For graph transformation task, we conduct experiments on two models, Interaction Networks [38] and NEC-DGT [31].

D.1 Molecule Generation Results.

As mentioned above, graph generation task could be very domain-specific, meaning that each domain has specific expectations over the generative tasks. Our first benchmark focuses on one of the most developed areas, molecular graph generation, which is motivated by drug and material discovery. For molecule generation task, we utilize the above mentioned self-quality based evaluation, where the validity, uniqueness and novelty are measured. We survey a list of state-of-the-art deep generative models on molecules and report the performance regarding validity, novelty, and uniqueness on two popular benchmark datasets (QM9 [44] and ZINC250K [45]) in the original paper as shown in Table 4. In Table 4, it is clearly to observe that the state-of-the-art models, such as MoFlow, GraphEBM, GraphDF, almost perform perfectly on the two common benchmarked datasets. As described in the following section, one key point to generate good molecular graphs is to handle the valency constraints. Some models utilize sequential generation, some utilize valency check, some design regularization, but overall, the best-performing models handle the valency constraint properly. However, it is not the end of the area. The molecule space being searched currently is small with very limited set of atoms and bonds and small size of molecules. Thus, benchmark datasets with larger molecules and molecules with more diverse atom and bond types are urgent to advance the field. From another perspective, it is important to generate molecules with desired properties which more domain-specific analyses and explorations could be done.

D.2 Baseline Models

GraphRNN [34]. GraphRNN represents graph generation as an auto-regressive process and builds an generative RNN model to generate nodes and edges sequentially.

GraphVAE [18]. GraphVAE represents each graph by its adjacency matrix and feature vectors and utilizes graph neural network to encode the graphs into a vector space. Then, the model learns the distribution of the graphs via a VAE setting which minimizes the distance between the latent distribution and Gaussian distribution. Finally, the model decodes the latent vectors to reconstruct graphs.

Table 4: Quantitative evaluation and comparison on molecular graph generation tasks by different deep generative models on graphs ("Valid." is short for validness. "Novel." is short for novelty. "Unique." is short for uniqueness.).

Method→	QM9			ZINC250K		
Dataset↓	Valid.	Novel.	Unique.	Valid.	Novel.	Unique.
GrammarVAE [107]	31.00%	**100.00%**	10.76%	30.00%	95.44%	9.30%
GraphVAE [18]	14.00%	**100.00%**	31.60%	61.00%	85.00%	40.90%
CGVAE [108]	**100.00%**	**100.00%**	99.82%	**100.00%**	94.35%	98.54%
GraphNVP[86]	74.30%	**100.00%**	94.80%	90.10%	54.00%	97.30%
GRF [109]	73.40%	**100.00%**	53.70%	84.50%	58.60%	66.00%
GraphAF[35]	**100.00%**	**100.00%**	99.10%	**100.00%**	88.83%	94.51%
CGSVAE [110]	34.90%	**100.00%**	-	96.60%	97.50%	-
JT-VAE [20]	**100.00%**	**100.00%**	99.80%	-	-	-
GCPN [93]	**100.00%**	**100.00%**	**99.97%**	-	-	-
MolecularRNN [36]	**100.00%**	**100.00%**	99.89%	-	-	-
MolGAN [19]	-	-	-	98.10%	94.10%	10.40%
MPGVAE [111]	-	-	-	91.00%	54.00%	68.00%
SCAT [112]	-	-	-	47.40%	92.00%	98.30%
MoFlow [32]	**100.00%**	98.03%	99.20%	**100.00%**	**100.00%**	**99.99%**
GraphEBM [33]	**100.00%**	97.01%	97.90%	99.96%	**100.00%**	98.79%
GraphDF [37]	**100.00%**	98.10%	97.62%	**100.00%**	**100.00%**	99.55%

GraphGMG [8]. GraphGMG first learns a node-level embedding of a given graph, then learns a probability distribution over possible outcomes for each generation step. During the generation process, the model sequentially connects nodes and edges to a new graph.

GrammarVAE [107]. GrammarVAE is one of the first deep generative models that learn to generative novel molecules with a string representation.

GraphVAE [18]. GraphVAE is a VAE-based graph generative models that generates graphs in an one-shot fashion.

CGVAE [108]. CGVAE is a VAE-based graph generative model that formulates the generation process as an iterative process.

GraphNVP [86]. GraphNVP first introduces the idea of invertible normalizing flow-based methods to molecular graph generation in an one-shot generation way.

GRF [109]. GRF introduces residual flows for molecular graph generation which circumvents the requirement of partitioning of the latent vector in GraphNVP.

GraphAF [35]. GraphAF takes one step further than GraphNVP to formulate the problem as a sequential generation problem.

CGSVAE [110]. CGSVAE is a VAE-based graph generative models that proposes a regularization method that encourages the model to generate valid molecules.

JT-VAE [20]. JT-VAE is motivated to explicitly model substructures in the generative models that introduces an extra junction tree encoder-decoder part which each node denotes a substructure rather than an atom in a molecule.

GCPN [93]. GCPN formulates molecular graph generation as a reinforcement learning problem where each state is a generation step, every step, it takes the action to connect two atoms and labels the edges by bond types. It stops when no atoms are connected.

MolecularRNN [36]. MolecularRNN follows the idea of GraphRNN and adopts it for the molecular graph generation task.

MolGAN [19]. MolGAN is a GAN-based molecular graph generation method that implements a GAN model to generate molecular graphs in an one-shot fashion.

MPGVAE [111]. MPGVAE designs a VAE-based which follows Graphite [113] and generate molecular graphs in an one-shot way.

SCAT [112]. SCAT takes a scattering transform and gaussianization as an encoder and utilizes a MLP as a decoder to generate novel molecular graphs in an one-shot way.

MoFlow [32]. MoFlow improves over GraphNVP by introducing a valency correction mechanism in the framework.

Table 5: Hyper-parameters for graph generation benchmark.

Method	Learning Rate	Epoch	Batch Size	# graphs
GraphRNN	3×10^{-3}	1,000	32	1,000
GraphVAE	1×10^{-3}	10	1	1,000
GraphGMG	1×10^{-3}	10	1	1,000

Table 6: Hyper-parameters for graph transformation benchmark. Due the capacity of our memory, for the graph transformation task, we sampled a subset from a few datasets for evaluation. The size of the subset depends on the graph size and total number of graphs contained in the dataset.

Dataset	Interaction Networks			NEC-DGT		
	Learning Rate	Epoch	#graphs	Learning Rate	Epoch	#graphs
AuthNet	1×10^{-2}	100	412	1×10^{-4}	500	412
Barab'asi-Albert Graphs	1×10^{-2}	100	1,000	1×10^{-4}	500	1,000
Brain-restingstate	1×10^{-2}	100	823	1×10^{-4}	500	823
Brain-emotion	1×10^{-2}	100	811	1×10^{-4}	500	811
Brain-grambling	1×10^{-2}	100	818	1×10^{-4}	500	818
Brain-language	1×10^{-2}	100	816	1×10^{-4}	500	816
Brain-motor	1×10^{-2}	100	816	1×10^{-4}	500	816
Brain-relational	1×10^{-2}	100	808	1×10^{-4}	500	808
Brain-social	1×10^{-2}	100	816	1×10^{-4}	500	816
Brain-wm	1×10^{-2}	100	812	1×10^{-4}	500	812
Scale-free	1×10^{-2}	100	250	1×10^{-4}	500	250
TwitterNet	1×10^{-2}	100	250	1×10^{-4}	500	250
N-body-charged	1×10^{-2}	100	150	1×10^{-4}	500	150
N-body-spring	1×10^{-2}	100	150	1×10^{-4}	500	150
ChemReact	1×10^{-2}	100	1,000	1×10^{-4}	500	1,000
IoTNet	1×10^{-2}	100	343	1×10^{-4}	500	343
MolOpt	1×10^{-2}	100	500	1×10^{-4}	500	500

GraphEBM [33]. GraphEBM is an energy-based generative model that utilizes Langevin Dynamics to sample novel molecules.

GraphDF [37]. GraphDF improves over GraphAF by learning discrete latent variables rather than continuous latent variables as in most of the Flow and VAE-based methods.

Interaction Network [38]. Physical domain is the target for Interaction Networks, the input of which is a graph that represents a system of objects and relations. Interaction Networks instantiates the pairwise interaction and compute its effects via a relational model. The effects are then aggregated and combined with the objects and external effects to generate the input for an object model, which predicts how the interactions and dynamics influence the objects.

NEC-DGT [31]. In NEC-DGT, the node and edge attributes of input graphs are inputted to the model. The model outputs node attributes and edges attributes of the generated target graphs via several blocks, which have edge and node translation paths co-evolved and combined by a graph regularization during training process.

D.3 Hyper-parameters

All experiments are conducted on a 64-bit machine with a 6 core Intel CPU i9-9820X, 32GB RAM, and an NVIDIA GPU (GeForce RTX 2080ti, 1545MHz, 11GB GDDR6). The detailed hyper-parameters can be found in Table 5 and Table 6. For the molecular graph generation benchmark, we take experiment results from the original reports.

E Tutorials

We provide data processors, evaluators, as well as visualizers which simplify the pipeline for graph generation and transformation, as shown in Fig. 4, 5 and 6, 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 4: 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 5: 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 6: Evaluation APIs.