Controlled Transformation of Text-Attributed Graphs

Nidhi Vakil
Department of Computer Science
University of Massachusetts Lowell
nvakil@cs.uml.edu

Hadi Amiri
Department of Computer Science
University of Massachusetts Lowell
hadi@cs.uml.edu

Abstract

Graph generation is the process of generating novel graphs with similar attributes to real world graphs. The explicit and precise control of granular structural attributes, such as node centrality and graph density, is crucial for effective graph generation. This paper introduces a controllable multi-objective translation model for text-attributed graphs, titled **C**ontrolled **G**raph **T**ranslator (CGT). It is designed to effectively and efficiently translate a given source graph to a target graph, while satisfying multiple desired graph attributes at granular level. Designed with an encoder-decoder architecture, CGT develops fusion and graph attribute predictor neural networks for controlled graph translation. We validate the effectiveness of CGT through extensive experiments on different genres of datasets. In addition, we illustrate the application of CGT in data augmentation and taxonomy creation, particularly in low resource settings.[1].

1 Introduction

Graph generation is the process of generating new graphs that satisfy the properties of existing real-world graphs (Erdös and Rényi, 1959; Barabási and Albert, 1999; You et al., 2018). Graph generation has extensive real world applications. These include generating new molecules that satisfy certain chemical properties for applications like drug discovery (Jin et al., 2018; Shi et al., 2019; Jin et al., 2020a), creating periodic graphs with repetitive patterns for new synthetic materials (Wang et al., 2022), generating synthetic social network for studying network dynamics and understanding complex social interactions (Pitas, 2016), generating program graphs for representing source codes using abstract syntax trees (Allamanis et al., 2018), and completing knowledge graphs (Melnyk et al., 2022; Zhou et al., 2023; Cao et al., 2023).

[1]Code and data are available at `https://clu.cs.uml.edu/tools.html`

Effective neural graph generation methods have been proposed. In particular, You et al. (2018) and Li et al. (2018) proposed to turn input graphs into sequences and generate graphs sequentially using two recurrent neural networks, which iteratively determine if and how new nodes and edges should be added to the partially generated graph. Several effective methods have been developed specifically for text graph generation (Koncel-Kedziorski et al., 2019; Jin et al., 2020b; Guo et al., 2020; Yao et al., 2020; Song et al., 2020; Saha et al., 2022; Melnyk et al., 2022; Han and Shareghi, 2022; Edwards et al., 2022). For example, Guo et al. (2020) proposed a variational autoencoder that generates graphs through disentanglement of node and edge features. The model implements three encoders to model the distribution of node, edge and graph features; and two decoders to jointly generate these features based on the resulting latent representations. Recently, Han and Shareghi (2022) proposed a low-resource approach to generate text from input graph by introducing several graph masking strategies to integrate both local and global information into the pre-trained language models. However, despite their effectiveness, existing works often lack explicit control over desired structural properties, or focus on a limited set of graph properties in the output space.

In this paper, we propose the **C**ontrolled **G**raph **T**ranslator (CGT), which transforms a given source graph into a new target graph that satisfies multiple, precisely-defined structural attributes. CGT implements an effective graph encoder-decoder model, augmented with a fusion and pre-trained structural attribute predictor. This predictor efficiently computes the attributes of graphs generated in real-time to explicitly guide the generation process. Specifically, the model enables translating a source graph into a target graph that not only satisfies the desired attributes but does so with granular control over the changes in its structural attributes.

The contribution of this paper are (a): a novel approach capable of controlling the transformation of text-attributed graphs and satisfying predefined structural attributes, and (b): a graph attribute predictor that explicitly guides the generation process by computing graph attributes in real-time.

We evaluate the efficacy of CGT across several domains including ontologies, citation networks, and molecular biology. It consistently outperforms existing baselines on these datasets. In addition, we show that CGT is capable of refining existing ontologies, such as WordNet (Miller, 1995), and generating new additional data points in low resource domains such as molecular biology, thereby improving the performance of downstream models developed for specific tasks. In addition, CGT is scalable to larger graphs.

2 Control Attributes

We provide a list of structural attributes that are crucial for controlled graph translation. These attributes, which have not been previously utilized in graph generation, add granularity, complexity and depth to the generation process, and impose explicit and precise control over the generation process:

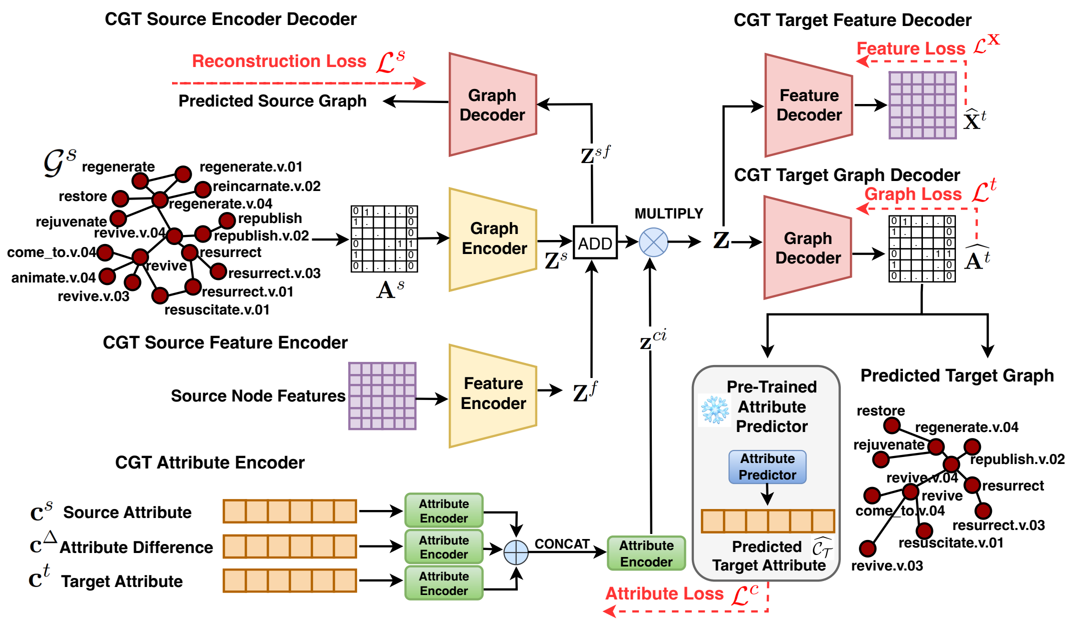
- **Number of nodes and edges**: These commonly-used and fundamental attributes define the scale and complexity of a graph.
- **Number of local bridges:** A local bridge is an edge that is not part of a triangle in the graph. Local bridges act as edges that transfer information between different parts of graphs.
- **Graph density:** The density of an undirected graph is computed as $\frac{e}{v(v-1)}$, where e is the number of edges and v is the number of nodes in the graph.
- **Edge connectivity:** Edge connectivity is the minimum number of edges that must be removed to disconnect the given graph.
- **Number of maximum cliques**: This attribute represents the number of maximum cliques within a graph. A maximum clique is a largest complete subgraph in a graph.
- **Diameter**: The diameter of a graph is the length of the shortest path between the most distanced nodes in the graph.
- **Treewidth min degree:** The treewidth of a graph is an integer that quantifies the degree to which a given graph deviates from a tree.
- **Transitivity:** Transitivity of a graph is the fraction of all possible triangles present in the graph. It is computed as $\frac{3\times|triangles|}{|triads|}$, where a "triad" is a set of two edges in the graph that share a common node.
- **Average closeness centrality:** The closeness of a node is its average distance to all other nodes in the graph or, in the case of disconnected graphs, to all other nodes in the connected component containing that node. For a given graph, we compute the average of closeness centrality scores of all nodes.
- **Average clustering coefficient:** The clustering coefficient of a node is the fraction of triangles that exist in its neighborhood (over all possible triangles). The clustering coefficient of a graph is the average of the clustering coefficient scores of its nodes.

These attributes reflect the complexity involved in transforming graphs through structural modifications. They enable precise control over the structures of generated graphs, making them suitable for specific applications. For example, attributes like number of nodes and edges, and local bridges are crucial in ontologies like WordNet, where the richness of linguistic connections, e.g. hypernyms and hyponyms, reflects the complexity and depth of semantic relationships. Edge connectivity and subgraph density are key in citation networks to model the influential works and inter-connectivity of research areas. In molecular datasets, transitivity can indicate the overall connectivity and potential stability of a compound, while clustering coefficient can help understand the compactness of molecular structures. In social networks, graph density is a crucial metric for community detection and analyzing the likelihood of forming tight-knit communities within graphs.

3 Controlled Graph Translator

Problem Statement Given a source graph $\mathcal{G}^s$, a source attribute vector $\mathbf{c}^s$, and a target attribute vector $\mathbf{c}^t$, we aim to translate $\mathcal{G}^s$ to a target graph $\mathcal{G}^t$ that satisfies specified desired attributes $\mathbf{c}^t$.

Solution Overview Given the source graph $\mathcal{G}^s = (V^s, E^s)$, we represent the adjacency matrix of $\mathcal{G}^s$ as $\mathbf{A}^s \in \mathbb{R}^{|V^s|\times|V^s|}$, such that $\mathbf{A}^s_{ij} = 1$ if there is an edge between node i and j, and $\mathbf{A}^s_{ij} = 0$ otherwise. Similarly, the adjacency matrix for the target graph $\mathcal{G}^t$ is represented as $\mathbf{A}^t \in \mathbb{R}^{|V^t|\times|V^t|}$.



CGT Source Encoder Decoder
Reconstruction Loss $\mathcal{L}^s$
Predicted Source Graph
Graph Decoder
$\mathcal{G}^s$
Graph Encoder
$\mathbf{A}^s$
CGT Source Feature Encoder
Source Node Features
Feature Encoder
$\mathbf{Z}^s$
$\mathbf{Z}^f$
$\mathbf{Z}^{sf}$
ADD
MULTIPLY
$\mathbf{z}^{ci}$
$\mathbf{Z}$
CGT Target Feature Decoder
Feature Loss $\mathcal{L}^X$
Feature Decoder
$\widehat{\mathbf{X}}^t$
CGT Target Graph Decoder
Graph Loss $\mathcal{L}^t$
Graph Decoder
$\widehat{\mathbf{A}}^t$
CGT Attribute Encoder
$\mathbf{c}^s$ Source Attribute
$\mathbf{c}^\Delta$ Attribute Difference
$\mathbf{c}^t$ Target Attribute
Attribute Encoder
CONCAT
Pre-Trained Attribute Predictor
Attribute Predictor
Predicted $\widehat{\mathcal{C}_T}$ Target Attribute
Attribute Loss $\mathcal{L}^c$
Predicted Target Graph

Figure 1: Architecture of Controlled Graph Translator (CGT). The model encodes the adjacency matrix ($\mathbf{A}^s$) of a source graph to obtain its structural representation $\mathbf{Z}^s$, which is integrated with the source features embeddings $\mathbf{Z}^f$ to obtain the source graph representation $\mathbf{Z}^{sf}$. It computes a control vector $\mathbf{z}^{ci}$ that effectively captures the attributes of the source and target graphs, and the desired changes in the source attributes for successful transformation. $\mathbf{z}^{ci}$ is integrated with $\mathbf{Z}^{sf}$ through element-wise multiplication to create final representation $\mathbf{z}$ as the input to the decoder. The decoder uses $\mathbf{z}$ to reconstruct the source graph features and generate the target graph. CGT uses a pre-trained attribute predictor to estimate attributes of the predicted graph and guide the generation process.

To simplify the translation process, we assume a maximum threshold for the number of nodes in the input and output, and use V to refer to both V^s and V^t. As shown in Figure 1, CGT encodes the adjacency matrix of $\mathcal{G}^s$ (e.g. a subgraph from WordNet) to obtain its structural representation $\mathbf{Z}^s$, and concurrently encodes the node features of $\mathcal{G}^s$ to obtain its content-based representation $\mathbf{Z}^f$. These two representations are combined to form the graph representation $\mathbf{Z}^{sf}$ (§3.1). Also, CGT computes a control vector $\mathbf{z}^{ci}$ that effectively captures the source and target attributes, and the desired attribute changes for successful translation, $\mathbf{c}^\Delta = \mathbf{c}^s - \mathbf{c}^t$. The control vector is integrated with $\mathbf{Z}^{sf}$ through element-wise multiplication to obtain the final latent representation $\mathbf{z}$, which encodes all the necessary information for transformation. The rationale for using element-wise multiplication is to allow the model to automatically adjust the influence of each feature for controlled graph transformation. The decoder then uses $\mathbf{z}$ to not only generate the source graph's attributes but also to generate the target graph (§3.2 and 3.3). A pre-trained attribute predictor (§3.4) assesses how well the predicted target graph matches the desired target attributes and use this feedback to guide the generation process.

3.1 Learning Representation

Given the non-Euclidean nature of graph data, we use a multilayer perceptron (MLP) to encode both structural information and textual (node) features of the source graph $\mathcal{G}^s$. While our framework is compatible with other neural networks such as graph neural networks (GNNs) or convolution neural networks (CNNs), we focus on MLP due to our comparative analyses.[2]

We add the structural ($\mathbf{Z}^s$) and feature ($\mathbf{Z}^f$) representations of the source graph to obtain its combined graph representation, $\mathbf{Z}^{sf}$. To ensure that $\mathbf{Z}^{sf}$ effectively captures the structure and content information of the source graph, we reconstruct the source graph from this representation. A low reconstruction loss indicates that all the key input information has been preserved in $\mathbf{Z}^{sf}$. We use MLP decoder to reconstruct the input source graph, $\mathcal{G}^s$. During training the model, we enforce this constraint by computing the loss between input source graph and predicted source graph as follows:

$$\ell^s_{ij} = \mathbf{A}^s_{ij}\log(P^s_{ij}) + (1-\mathbf{A}^s_{ij})\log(1-P^s_{ij}), \quad (1)$$

[2]We analyzed several types of GNNs and found that they tend to oversmooth the representations, which blurs distinctive features among nodes and compromises the reconstruction performance. Similarly, CNN was not effective on non-Euclidean data such as graphs, which, despite CNN, are order insensitive.

$$\mathcal{L}^s = \frac{1}{|v|^2}\sum_{i=1}^{v}\sum_{j=1}^{v}\ell_{ij}^s, \tag{2}$$

where P_{ij}^s is the probability of an edge between nodes i and j in $\mathcal{G}^s$, and $\mathbf{A}_{ij}^s$ is the ground truth.

3.2 Integrating Control Attributes

To encode the source and target attribute vectors, $\mathbf{c}^s$ and $\mathbf{c}^t$, and their difference vector $\mathbf{c}^\Delta$, we process each attribute vector independently using an MLP. We obtain the representation for $\mathbf{c}^s$ as:

$$\mathbf{h}_{\mathbf{c}^s}^{(m+1)} = \texttt{ReLU}\Big(\mathbf{W}_{\mathbf{c}^s}\mathbf{h}_{\mathbf{c}^s}^{(m)} + \mathbf{b}_{\mathbf{c}^s}\Big), \tag{3}$$

where $\mathbf{W}_{\mathbf{c}^s}$ and $\mathbf{b}_{\mathbf{c}^s}$ are learnable parameters, $h_{\mathbf{c}^s}^0 = \mathbf{c}^s$, and m is set from $\{0, 1\}$. Similarly, we learn latent representations for $\mathbf{c}^t$ with $h_{\mathbf{c}^t}^0 = \mathbf{c}^t$ and $\mathbf{c}^\Delta$ with $h_{\mathbf{c}^\Delta}^0 = \mathbf{c}^s - \mathbf{c}^t$. The resulting representations from each MLP are concatenated and processed with fully connected layer to produce the control vector representation $\mathbf{z}^{ci}$. This vector effectively guides the transformation process by influencing how the graph's structure and attributes should adjust to meet specific target criteria.

3.3 Target Graph Generation

The source representations $\mathbf{Z}^{sf}$ and the control vector $\mathbf{z}^{ci}$ from attribute encoder are integrated through element-wise multiplication to obtain the input to the decoder, $\mathbf{z}$, to generate the target graph. In CGT, the target graph decoder predicts the target graphs using a fully connected decoder layer, where we use the binary cross entropy loss between target graph and predicted target graph as follows:

$$\ell_{ij}^t = \mathbf{A}_{ij}^t \log(P_{ij}^t) + (1 - \mathbf{A}_{ij}^t)\log(1 - P_{ij}^t), \tag{4}$$

$$\mathcal{L}^t = \frac{1}{|v|^2}\sum_{i=1}^{v}\sum_{j=1}^{v}\ell_{ij}^t, \tag{5}$$

where P_{ij}^t is the predicted probability of an edge between nodes i and j in the target graph $\mathcal{G}^t$, and $\mathbf{A}_{ij}^t$ is the ground truth.

In addition, we use $\mathbf{z}$ to predict target features using another fully connected decoder layer, where we minimize the mean square error (MSE) between predicted target features $\widehat{\mathbf{X}}^t$ and target features $\mathbf{X}^t$:

$$\mathcal{L}^{\mathbf{X}} = \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}\Big(\mathbf{X}^t - \widehat{\mathbf{X}}^t\Big)^2. \tag{6}$$

3.4 Control Attribute Predictor

To enable end-to-end training, independently pre-train and freeze an attribute predictor that predicts structural (control) attributes of a graph from its adjacency matrix. We pre-train the model using training ($\mathbf{A}^s$, $\mathbf{c}^s$) and ($\mathbf{A}^t$, $\mathbf{c}^t$) pairs. This model allows efficient and real-time computation of graph attributes during end-to-end training, while keeping the model differentiable. We optimize the target graph transformation with attribute loss as follows:

$$\widehat{\mathbf{c}^t} = \mathbf{W}_c\mathbf{h}^{last} + \mathbf{b}_c \tag{7}$$

$$\mathbf{h}^{(m+1)} = \texttt{ReLU}\Big(\mathbf{W}_c\mathbf{h}^{(m)} + \mathbf{b}_c^{(m)}\Big), \tag{8}$$

where $\mathbf{h}^{(0)} = \texttt{flatten}(\widehat{\mathbf{A}^t})$; $\texttt{flatten}$ convert $\widehat{\mathbf{A}^t}$ into a vector, $\widehat{\mathbf{c}^t}$ denotes predicted attribute vector, and $\mathbf{W}_c$ and $\mathbf{b}_c$ are learnable parameters. Here, $\mathbf{h}^{last}$ indicates the representation from the final MLP layer. We use MSE loss to train the attribute index predictor model as follows:

$$\mathcal{L}^c = \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}(\mathbf{c} - \widehat{\mathbf{c}})^2, \tag{9}$$

where $\mathbf{c}$ is the ground truth attribute vector (either from source or target in training data), $\widehat{\mathbf{c}}$ denotes predicted attribute vector, and $|\mathcal{D}|$ denotes total number of graphs in training.

3.5 Objective Function

We obtain the final loss function to train CGT by linearly combining the above four loss functions:

$$\mathcal{L}^{total} = \alpha\mathcal{L}^t + \beta\mathcal{L}^s + \gamma\mathcal{L}^{\mathbf{X}} + \delta\mathcal{L}^c \tag{10}$$

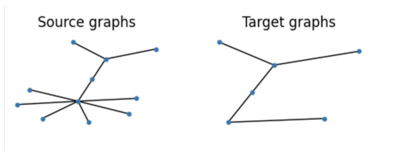
where α, β, γ, and δ are the weights for each loss.

4 Experiments

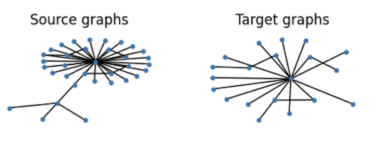
4.1 Datasets

WordNet (Miller, 1995): a dataset of English words where nouns, verbs, adjectives and adverbs are grouped into sets of synonyms based on their meanings. We use hypernyms, hyponyms, meronyms, and holonyms relationships in WordNet to create different WordNet graphs.

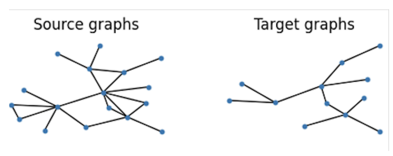
Ogbn-arxiv (Hu et al., 2020): a citation network between arxiv papers in computer science, where each node is a paper and an edge represents a citation from one paper to another. In addition, each paper contains an embedding vector obtained from the average of the words present in the title and abstract of the paper.



(a) WordNet



(b) Ogbn-Arxiv



(c) Citeseer

Figure 2: Source-Target pairs used to train all models. Source graphs are constructed by extracting k-hop neighbor for each node. Target graphs are created by randomly removing $n\%$ of edges from the source graphs (such that no more than one connected component remains after edge removal). In addition, source-target pairs are flipped to allow learning transformations both from dense to sparse and sparse to dense structures.

	Train	**Val**	**Test**
WordNet	52,675	2,926	2,927
Citeseer	1,406	78	79
Arxiv	47,538	2,641	2,641
MUTAG	169	10	9
MOLBACE	1,323	74	74

Table 1: Dataset statistics in terms of the number of source-target graph pairs in each data split.

Citeseer (Kipf and Welling, 2017a): a citation network of scientific articles, where nodes are papers and edges indicate citations between them.

MUTAG (Morris et al., 2020): a molecular dataset where each graph represents a chemical compound and classified as whether the given molecule have mutagenic effect on specific gram negative bacaterium.

MOLBACE (Hu et al., 2020): a molecular dataset where each graph represents a chemical compound.

Table1 shows the statistics of these datasets.

4.2 Generating Source-Target Pairs

To create source-target graph pairs from the datasets, we consider the k-hop neighbors of each node in the graph to create source subgraphs. We randomly remove a maximum of $n\%$ edges from each source subgraph such that no more than one connected component remains after edge removal. We flip these source-target pairs to allow the model learns transforming graph from sparse to dense and vice-versa. Figure 2 shows examples source-target pairs with $n = 50\%$ and $k = 3$.

4.3 Settings

we use the Networkx package (Hagberg et al., 2008) to calculate the graph attributes in §2. We set the maximum number of nodes to $|V| = 50$ in experiments and zero pad adjacency matrix as necessary. This threshold is appropriate for GNNs due to the nature of how GNNs process graph data, especially when considering the common practice of sampling 1-2 hop neighbors form localized subgraphs for nodes. We set the number of hops to to $k = 2$ for WordNet and Ogbn-Arxiv datasets and to $k = 3$ for Citeseer (due to its smaller size). We consider a batch-size of 4,096. We consider maximum number of 1000 training iteration for Citeseer and 200 iterations for Ogbn-Arxiv and WordNet datasets. For the CGT(CNN) encoder, we used two layers of CNN with kernel size of 5 and 32, 64 channels respectively. For the decoder, we used two layers of CNN with 64,32 channels respectively. For We run all experiments on a single A100 40GB GPU.

4.4 Evaluation Metrics

We use mean absolute difference (MAD↓) and graph edit distance (GED↓) for evaluation. MAD computes the absolute difference between the attributes of predicted graphs and their corresponding target graphs. We average these absolute differences for the dataset. GED is a graph similarity measure which is similar to Levenshtein distance for strings. It gives the minimum cost for transforming one graph to another. Due to the extensive time complexity of GED computation, we only evaluate this metric for the test graphs with 10 or less nodes.

4.5 Baselines

For fair evaluation, we incorporate our control target attributes to all baseline models except GraphRNN which is a free generative model.

GraphRNN (You et al., 2018): generates graph iteratively by training on a representative set of graphs using breath first search of nodes and edges and implements node and edge RNNs to generate target graphs. GraphRNN is not a controlled generation approach.

	WordNet		Citeseer		Ogbn-Arxiv		MUTAG		MOLBACE		Average	
	GED	**MAD**	**GED**	**MAD**	**GED**	**MAD**	**GED**	**MAD**	**GED**	**MAD**	**GED**	**MAD**
GraphRNN (You et al., 2018)	4.02	3.26	5.42	5.05	5.36	4.80	–	1.71	–	3.81	4.93	3.73
GT-GAN (Guo et al., 2023)	7.16	2.74	5.14	2.73	7.81	3.24	–	2.64	–	5.26	6.70	3.32
GenStat (Zahirnia et al., 2024)	3.86	4.11	5.62	5.35	5.58	5.53	–	4.14	–	6.89	5.02	5.20
EDGE (Chen et al., 2023)	4.73	3.87	5.50	5.50	6.39	5.60	–	2.34	–	3.51	5.54	4.16
CGT (CNN)	3.39	1.17	5.92	2.53	5.18	2.39	–	1.77	–	**2.36**	4.83	2.54
CGT (MLP)	**1.67**	**0.56**	**4.20**	**2.47**	**3.30**	**1.59**	–	**0.75**	–	3.25	**3.05**	**1.72**

Table 2: Overall performance across datasets. Average mean absolute difference (MAD ↓) is the average of absolute mean error in satisfying target attributes. Lower is better. Here, "–" in GED (↓) indicates missing values as there were no predicted graphs with less than 10 nodes to calculate GED. Variance is reported in Appendix B

	Hyponyms		Hypernyms		P.Holonyms		P.Meronyms		Entailments		Average	
	GED	**MAD**	**GED**	**MAD**	**GED**	**MAD**	**GED**	**MAD**	**GED**	**MAD**	**GED**	**MAD**
GraphRNN (You et al., 2018)	3.68	3.15	3.73	3.15	4.71	3.35	3.71	2.97	4.25	0.74	4.02	2.67
GT-GAN (Guo et al., 2023)	7.23	2.77	3.26	2.52	9.51	3.05	9.10	2.85	11.0	1.81	8.02	2.60
GenStat (Zahirnia et al., 2024)	3.24	4.12	3.24	4.12	3.54	4.87	4.00	4.15	**2.33**	**0.48**	**3.27**	3.54
EDGE (Chen et al., 2023)	4.08	3.80	4.15	3.81	5.06	4.36	5.56	4.28	4.33	1.22	4.63	3.49
CGT (CNN)	2.92	1.02	3.01	1.10	–	3.73	–	3.08	7.50	1.46	4.47	2.08
CGT (MLP)	**1.20**	**0.47**	**1.28**	**0.49**	**3.32**	**1.21**	**2.81**	**0.95**	11.5	2.32	4.02	**1.08**

Table 3: Performance across different types of WordNet relationships. Lower GED and MAD indicates better performance. P. stands for Part and "–" in GED indicates missing values as there were no predicted graphs with less than 10 nodes to calculate GED.

Deep Graph Translation (GT-GAN) (Guo et al., 2023): is a generative adversarial network for graph transformation. It employs a graph convolution neural network as graph discriminator conditioned on the input graph.

EDGE (Chen et al., 2023): is a diffusion based generative model which iteratively removes edges to create completely disconnected graph and uses decoder to iteratively reconstruct the original graph. It explicitly uses node degrees and adjacency matrix to satisfy the graph statistics of the generated graphs similar to the training graphs.

GenStat (Zahirnia et al., 2024): learns the latent adjacency matrix conditioned on graph level statistics, and decodes it to recreate statistics and use it to generate graphs.

4.6 Main Results

Table 2 shows the overall performance of models across datasets. The corresponding attribute-level performance is reported in Appendix B. The results show that CGT outperforms all the baselines across datasets except in Wordnet (Entailments) dataset; we attribute this to small number of data points in the graph. GraphRNN does not generate graphs conditionally dependent on attributes constraints leading to generation of graphs different from target graphs, and results in greater error rate compared to CGT. EDGE performs better compare to GenStat as EDGE generates graph based on the degree distribution similar to train data points by explicitly modeling adjacency matrix, while GenStat generate graphs using graph statistics but implicitly generating adjacency matrix from latent representations. CGT (MLP) and CGT (CNN) provides the granular control over the graph attributes during transforming to get the desired graph. As evident from the results, the mean average difference (MAD ↓) between the attribute values in predicted and target graphs is small, indicating the magnitude of control over desired attributes during generation.

Table 3 in the hyponyms and hypernyms categories, CGT(MLP) achieves the best GED and MAD scores. CGT(CNN) also does well but not as strong as the MLP version. In part holonyms and part meronyms, CGT(MLP) leads in GED and shows exceptional performance improvements in MAD over the CNN version and other models. The Entailments category is dominated by GenStat with the lowest GED and MAD, indicating its strength in handling this specific type of relationship, although CGT(MLP) did not perform as well here. The Average scores across all categories show that CGT(MLP) and CGT(CNN) have strong overall capabilities, with the MLP version providing the best average MAD and GraphRNN showing solid average GED.

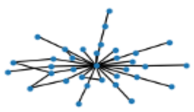
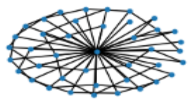
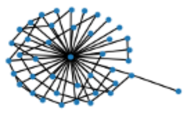
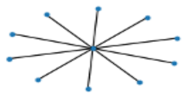
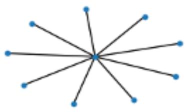
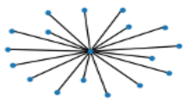
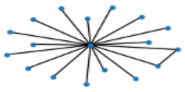
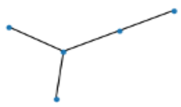
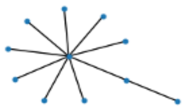
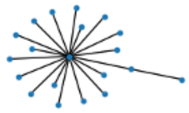
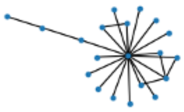
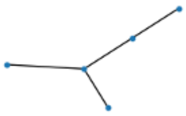
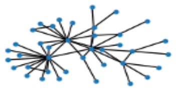
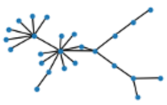
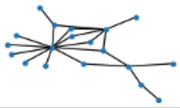
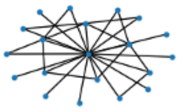
			CGT		GraphRNN	
Dataset	**Source**	**Target**	**Predicted**	**MAD↓**	**Predicted**	**MAD↓**
WordNet				1.09		10.66
				0.42		4.44
Ogbn - Arxiv				1.20		5.03
				1.68		2.98

Table 4: Example of the source, target, and predicted graphs for WordNet and Ogbn-Arxiv by CGT (MLP) and GraphRNN. Here, MAD indicates the difference between predicted graph and target graph.

Table 4 visualizes example source, target, and predicted graphs. CGT learns the attributes from diverse pool of source graphs and can generate target graphs that satisfy control attributes by effectively capturing the changes required to achive target graphs. In comparison, GraphRNN generates much sparser graph that target, resulting in higher MAD score.

4.6.1 Attribute Index Predictor

We pre-trained the attribute predictor as described in §3.4. Table 5 shows an exceptionally low MSE for the predictor across all the datasets. Such effective estimations allows CGT to precisely predict the attributes of its predicted target graphs.

Dataset	Mean Square Error↓
Ogbn-Arxiv	0.00000209
Wordnet	0.00000289
Citeseer	0.00073484
MUTAG	0.00201862
MOLBACE	0.00004967

Table 5: MSE of pre-trained attribute predictor.

5 Discussion

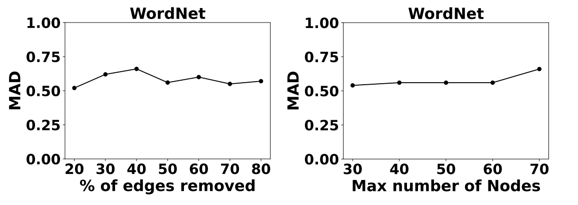
Ablation Study Several factors affect training CGT, see (10), especially, the latent representation $\mathcal{Z}$, which is used to generate target graphs, features, and graph attributes. Table 6 shows changes in the error rate while generating target graphs for all the datasets across objective functions. The results show that, overall, each of the terms in the objective function contribute to learning better graph generation and transformation. Ignoring feature reconstruction (w/o feature loss) results in increase in the error. This indicates the importance of using features to guide the model in learning better graph representations. Removing feature loss increases the error in Arxiv, WordNet, Citeseer and MUTAG, perhaps because the model loses a critical guide that helps it understand and reconstruct the complex relationships and properties encoded in the features. On the other hand, it decreases the error rate in MOLBACE. This is perhaps because this dataset might be less dependent on node features, or the graph structure itself might provide sufficient information for the generation. Ignoring graph attributes (w/o attribute loss) also increases the error indicating the importance of informing model about deviation in desired attributes.

Effect of Pruning on CGT As described in §4.2, we remove a maximum of $n\%$ of edges from the source graph to obtain the source-target pairs. To investigate the effect of varying degrees of pruning, we train CGT independently for values of $n \in [20\% - 80\%]$ with step size of 10%. Figure 3(a) shows the ability of CGT in accurate generation of target graphs at different levels of edge removal; the graph with greater percentage of edges removed indicates the highest difference between source-target pairs. Figure 3(a) shows as more edges are removed to create the source-target graph pairs, MAD increases, indicating the challenges of satisfying significantly different control attributes. We note MAD improves between 40% to 70% edge

Loss Term	Ogbn-Arxiv	WordNet	Citeseer	MUTAG	MOLBACE
CGT (MLP)	1.59	0.56	2.47	0.75	3.25
w/o Feature Loss	1.65 (↑)	0.61 (↑)	2.50 (↑)	1.42 (↑)	2.44 (↓)
w/o Attribute Loss	1.67 (↑)	0.65(↑)	2.51 (↑)	1.13 (↑)	1.91 (↓)

Table 6: Ablation analysis. MAD error of CGT indicates the full model, "w/o" indicates the use of overall objective function without a particular loss term. (↑) indicates increase in error and (↓) indicates decrease in error.

removal. This could be because the pruning might help by eliminating redundant or less important connections within the graph.



(a) (b)

Figure 3: (a) shows the MAD error rate at different levels of edge removal on WordNet. As the percentage of removed edges increases, the source and target graph become significantly different and accurate prediction of the target graphs becomes more challenging. (b) shows the MAD error rate on larger graphs.

Effect of Number of Nodes To understand the strength and limitation of CGT, we create snapshots of multiple datasets from WordNet such that each dataset contains graphs with a maximum threshold for node size in $V \in [30 - 70]$. Figure 3(b) shows that the error rate increases for larger graphs. This is because, in larger graphs, important interactions can occur between distant nodes, making it difficult for models, especially MLP that primarily aggregate information, to accurately learn and predict these distant relationships.

Data Augmentation Data augmentation is crucial for low resource datasets. We augment graph data points for two of our smaller datasets, MUTAG and MOLEBACE, and evaluate its effect on their held-out dataset. We augment the data by creating new graphs through CGT by making small Gaussian changes to their attributes to generate target attributes and create target graphs. To enable data augmentation, we assume each augmented graph has the same label as its corresponding source graph. To determine the effect of augmentation, we use a binary graph classification model to compare two settings: (a): using the original train data without any augmentation and (b): with data augmentation using CGT. Table 7 shows a major accuracy gain of 10.5 points on a smaller MUTAG dataset and a gain of 4.6 points on relatively larger

Data Augmentation	MUTAG	MOLBACE
	Accuracy↑	
Original Train set	78.95	59.21
+ CGT augment	**89.47**	**63.82**

Table 7: Gain in Performance on the held-out set after augmenting the original train set with additional data points generated using CGT.We used Graph Neural Network (GNN) for the property classification task.

MOLBACE dataset. The results indicate that data augmentation obtained through CGT can improve the performance of the low resource dataset.

Ontology Refinement Wordnet is a useful linguistic resource, but it suffers from missing connections between its words. We illustrate that CGT can detect missing edges in WordNet for ontology refinement. Figure 4 shows the result of an experiment, where the attributes of the source graph (blue nodes) is given as target attributes to CGT. Therefore, CGT has to re-generates the source. Figure 4 shows a group of nodes that appear with high confidence in the generated graph. Many of these edges indicate genuine relations missing from WordNet, e.g. CGT confidently links "ethnic group" to "egyptian" in the hypernym graph.

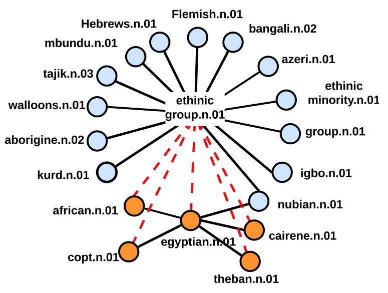


Figure 4: Ontology refinement by adding missing edges using CGT. Solid lines indicates existing connections in WordNet and dotted lines indicates edges generated by CGT originally missing from WordNet.

6 Related Work

Recent surveys (Faez et al., 2021; Zhu et al., 2022) shows that existing deep graph generation models uses various methods to generate graphs. Auto-regressive methods (You et al., 2018) generates the graph sequentially either node by node or edge by edge; Auto-encoder graph generators utilizes the latent space variables to generate graphs; Reinforcement learning methods to induce the desired properties into the graph; flow based learning methods which learns the mapping from the given graph distributions to mostly Gaussian distribution for learning data likelihood. GraphRNN (You et al., 2018) is a deep auto-regressive model with two Recurrent Neural Network (RNN) our of which Graph-level RNN maintains the state of the graph generated so far while edge-level RNN determines if an edge should be present between the current node and all the previously generated node by modeling the decision as multi-variate or univariate Bernoulli distribution. Li et al. (2018) proposes a sequential graph generation process by taking sequence of decisions whether to add a new node or not; and to decide which edges should be added and repeat this process till the model decides not to add any edge or node to terminate the generation.

There is extensive literature on generating new molecules structure using graph generation modeling. Jin et al. (2018) proposes JT-VAE which generates the molecules as an unconditional graph generation process by adopting small size motif sequence generating techniques based on Variation Auto Encoder (VAE). JT-VAE uses small size motifs as building block which degrades its performance for for generating larger molecules. To address this issue, HierVAE (Jin et al., 2020a) proposes to generate molecular graphs using significantly larger motifs where the encoder produces representation for each molecule in a fine-to-course fashion i.e. from atoms to connected motifs and decoder follows auto-regressive course-to-fine fashion to complete the molecule generation.

Kipf and Welling (2016) proposes generation model VGAE mainly for unsupervised learning on graphs based on VAE where given the graph,adjacency matrix and node features, VGAE infers latent matrix based on two-layer Graph Convolution Network (GCN) (Kipf and Welling, 2017b). The graph decoder is a simple inner product of latent variables. The main limitation of VGAE is that it works on single graph. To over come this limitation, Simonovsky and Komodakis (2018) proposes VAE based generative model that learns from a dataset of graphs where GCN encoder embeds the graphs to the continuous latent space and decoder outputs a probabilistic fully-connected graph with predefined maximum size for the graph. Guo et al. (2020) proposes NED-VAE framework based on node and edge disentanglement learning based on VAE where it has three encodes where each of them models the distribution of node, edge and graph features; and two decoders which jointly generates the node and edge features based on the latent representations. Wang et al. (2022) proposes Periodical-Graph Disentangled Variational Autoencoder (PGD-VAE) model for generating periodic graphs by disentangling the representations of local and global patterns in periodic graphs.

Melnyk et al. (2022) propose a model to generate a knowledge graph from input text where firstly entities are extracted as nodes using T5 language model (Raffel et al., 2020) and then the relations are classified using a classification head to build edges between the nodes using the node features. Song et al. (2020) proposes a model to generate text from graph by leveraging richer training signals like triple relation in which graph is broken into set of triples and is reconstructed to guide model to preserve input information using different aspects of input graphs. Saha et al. (2022) proposed a model to generate explanation graphs in an end-to-end manner using contrastive learning methods to improve on structural constrains and semantically correctness. Wang et al. (2021) proposes a model to generate long text from the short input text in a two stage approach to generate relatively more coherent long text, first by building document level directed semantic graph and the selecting a path that maximize subgraph matching for text generation.Han and Shareghi (2022) proposes a low-resource method to generate text from input graph by introducing several graph masking strategies to inject local and global information into the pre-trained language models. Other research in graph to text generation includes (Jin et al., 2020b; Yao et al., 2020; Koncel-Kedziorski et al., 2019)

7 Conclusion and Future Work

We introduce a novel approach for controlled transformation of text-attributed graphs. The model can control any number of structural attributes and implements a graph attribute predictor that explicitly guides the generation process by computing graph attributes in real-time. In future, we will extend our approach to generate diverse outputs, and investigate meta-learning approaches and minimal training for graph generation.

Limitations

The proposed model have been tested on graphs with maximum number of 80 nodes and performance may degrade as the graph size significantly increases. In addition, the proposed approach may require considerable tuning and adaptation to work effectively with directed graphs, weighted edges, or graphs with heterogeneous node types. The potential focus and solution is using efficient sparse matrix representation and factorization techniques, optimizing model architecture to reduce computational overhead, investigating more efficient graph sampling techniques, and building on inductive capability of GNNs.

References

Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. 2018. Learning to represent programs with graphs. In *International Conference on Learning Representations*.

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

Pengfei Cao, Yupu Hao, Yubo Chen, Kang Liu, Jiexin Xu, Huaijun Li, Xiaojian Jiang, and Jun Zhao. 2023. Event ontology completion with hierarchical structure evolution networks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 306–320. Association for Computational Linguistics.

Xiaohui Chen, Jiaxing He, Xu Han, and Liping Liu. 2023. Efficient and degree-guided graph generation via discrete diffusion modeling. In *International Conference on Machine Learning*, pages 4585–4610. PMLR.

Carl Edwards, Tuan Lai, Kevin Ros, Garrett Honke, Kyunghyun Cho, and Heng Ji. 2022. Translation between molecules and natural language. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 375–413.

P Erdös and A Rényi. 1959. 2017-10-20t13:47:06.000+0200. *Publicationes Mathematicae Debrecen*, 6:290–297.

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

Xiaojie Guo, Lingfei Wu, and Liang Zhao. 2023. Deep graph translation. *IEEE Trans. Neural Networks Learn. Syst.*, 34(11):8225–8234.

Xiaojie Guo, Liang Zhao, Zhao Qin, Lingfei Wu, Amarda Shehu, and Yanfang Ye. 2020. Interpretable deep graph generation with node-edge co-disentanglement. In *26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2020*, pages 1697–1707. Association for Computing Machinery.

Aric Hagberg, Pieter Swart, and Daniel S Chult. 2008. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Jiuzhou Han and Ehsan Shareghi. 2022. Self-supervised graph masking pre-training for graph-to-text generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4845–4853, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2018. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pages 2323–2332. PMLR.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2020a. Hierarchical generation of molecular graphs using structural motifs. In *International conference on machine learning*, pages 4839–4848. PMLR.

Zhijing Jin, Qipeng Guo, Xipeng Qiu, and Zheng Zhang. 2020b. GenWiki: A dataset of 1.3 million content-sharing text and graphs for unsupervised graph-to-text generation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2398–2409, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders.

Thomas N. Kipf and Max Welling. 2017a. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Thomas N Kipf and Max Welling. 2017b. Semi-supervised classification with graph convolutional networks.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Pa-*

pers), pages 2284–2293, Minneapolis, Minnesota. Association for Computational Linguistics.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. 2018. Learning deep generative models of graphs.

Igor Melnyk, Pierre Dognin, and Payel Das. 2022. Knowledge graph generation from text. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1610–1622, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. 2020. Tudataset: A collection of benchmark datasets for learning with graphs. *CoRR*, abs/2007.08663.

Ioannis Pitas. 2016. *Graph-based social media analysis*. CRC Press.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Swarnadeep Saha, Prateek Yadav, and Mohit Bansal. 2022. Explanation graph generation via pre-trained language models: An empirical study with contrastive learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1190–1208, Dublin, Ireland. Association for Computational Linguistics.

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. 2019. Graphaf: a flow-based autoregressive model for molecular graph generation. In *International Conference on Learning Representations*.

Martin Simonovsky and Nikos Komodakis. 2018. Graphvae: Towards generation of small graphs using variational autoencoders. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I 27*, pages 412–422. Springer.

Linfeng Song, Ante Wang, Jinsong Su, Yue Zhang, Kun Xu, Yubin Ge, and Dong Yu. 2020. Structural information preserving for graph-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7987–7998, Online. Association for Computational Linguistics.

Shiyu Wang, Xiaojie Guo, and Liang Zhao. 2022. Deep generative model for periodic graphs. *Advances in Neural Information Processing Systems*, 35.

Ziao Wang, Xiaofeng Zhang, and Hongwei Du. 2021. Building the directed semantic graph for coherent long text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2563–2572, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shaowei Yao, Tianming Wang, and Xiaojun Wan. 2020. Heterogeneous graph transformer for graph-to-sequence learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7145–7154.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. 2018. GraphRNN: Generating realistic graphs with deep auto-regressive models. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5708–5717. PMLR.

Kiarash Zahirnia, Yaochen Hu, Mark Coates, and Oliver Schulte. 2024. Neural graph generation from graph statistics. *Advances in Neural Information Processing Systems*, 36.

Wentao Zhou, Jun Zhao, Tao Gui, Qi Zhang, and Xuanjing Huang. 2023. Inductive relation inference of knowledge graph enhanced by ontology information. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 6491–6502. Association for Computational Linguistics.

Yanqiao Zhu, Yuanqi Du, Yinkai Wang, Yichen Xu, Jieyu Zhang, Qiang Liu, and Shu Wu. 2022. A survey on deep graph generation: Methods and applications. In *Learning on Graphs Conference*, pages 47–1.

A Effective Graph Encoder

We use CNNs to capture local and global representations while encoding the graph structure. This helps CGT to accurately transform source graphs while keeping the original representation intact. Figure 5 shows three input graphs and their reconstructed graphs for Citeseer, Wordnet and Ogbn-Arxiv respectively. A CNN encoder-decoder model leads to minimal reconstruction error.

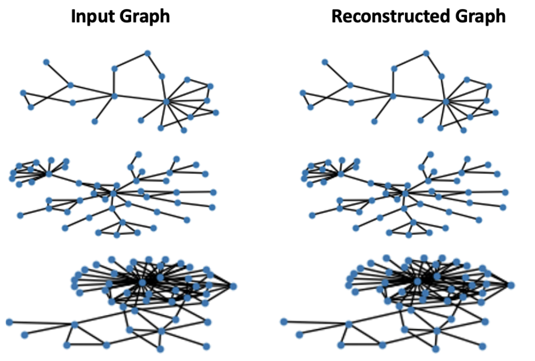


Figure 5: Outputs of CNN auto-encoder. Left shows the input graph and right shows the reconstructed output graph from Citeseer (top), Wordnet (Middle), and Ogbn-Arxiv (Bottom).

B Detailed Results

Detailed results for all the datasets considering each attribute is shown in Table 8 and 9. CGT (MLP) outperforms all the baselines for each attribute.

We report the mean and standard variance for all the datasets for 3 runs in Table 10 and 11.

	WordNet				Citeseer				Ogbn-Arxiv			
	CGT (CNN)	**CGT (MLP)**	**Graph RNN**	**GT-GAN**	**CGT (CNN)**	**CGT (MLP)**	**Graph RNN**	**GT-GAN**	**CGT (CNN)**	**CGT (MLP)**	**Graph RNN**	**GT-GAN**
Density	0.05	0.03	0.13	0.08	0.11	0.16	0.13	0.05	0.06	0.05	0.13	0.06
Edges	3.60	1.56	9.85	9.70	11.2	8.54	20.6	13.6	11.9	6.93	21.1	16.5
Nodes	3.27	1.30	9.47	7.38	5.53	5.90	13.4	5.64	5.09	2.72	12.2	6.61
Node Connectivity	0.02	0.00	0.00	0.01	0.01	0.08	0.00	0.00	0.01	0.01	0.01	0.01
Average Clustering	0.04	0.04	0.03	0.16	0.18	0.11	0.16	0.17	0.18	0.14	0.19	0.21
Closeness Centrality	0.04	0.03	0.09	0.04	0.12	0.16	0.13	0.06	0.07	0.07	0.12	0.08
Local Bridges	3.18	1.87	9.00	6.07	3.82	5.04	8.08	3.92	3.39	3.49	6.87	3.97
Transitivity	0.02	0.02	0.01	0.10	0.17	0.11	0.13	0.14	0.11	0.09	0.13	0.12
Edge Connectivity	0.00	0.00	0.00	0.01	0.01	0.08	0.01	0.01	0.01	0.01	0.01	0.01
Cliques	3.35	1.44	9.43	7.54	6.32	6.76	14.5	6.45	5.50	3.87	13.6	8.2
Treewidth Min Degree	0.20	0.19	0.26	1.16	1.00	0.74	1.22	1.13	0.95	0.75	1.3	1.36
Diameter	0.35	0.35	0.89	0.69	1.88	2.01	2.21	1.56	1.44	1.03	1.81	1.75
MAD	1.17	0.56	3.26	2.74	2.53	2.47	5.05	2.72	2.39	1.59	4.80	3.24

Table 8: Performance of CGT compared with Graph-RNN and GT-GAN on Citeseer, WordNet, Ogbn-Arxiv Datasets. Here, each number indicates the absolute mean error of graph attributes between the predicted graph and the target graph. Lower the value, better the performance.

	MUTAG				MOLBACE			
	CGT (CNN)	**CGT (MLP)**	**Graph-RNN**	**GT-GAN**	**CGT (CNN)**	**CGT (MLP)**	**Graph-RNN**	**GT-GAN**
Density	0.07	0.02	0.04	0.06	0.02	0.11	0.03	0.03
Edges	4.89	1.67	4.56	7.88	7.31	8.34	12.0	16.9
Nodes	4.00	1.22	3.11	5.62	5.58	7.51	10.4	11.6
Node Connectivity	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Average Clustering	0.02	0.00	0.00	0.07	0.01	0.01	0.00	0.07
Closeness Centrality	0.07	0.05	0.08	0.09	0.03	0.14	0.04	0.11
Local Bridges	4.22	2.44	4.56	4.38	5.11	8.24	7.15	9.97
Transitivity	0.03	0.00	0.00	0.10	0.01	0.02	0.00	0.10
Edge Connectivity	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Cliques	5.11	1.67	4.56	8.12	7.26	8.66	12.0	15.9
Treewidth Min Degree	0.44	0.11	0.67	1.50	0.53	0.18	0.47	1.95
Diameter	2.44	1.89	3.00	3.88	2.47	5.85	3.42	6.32
MAD	1.77	0.75	1.71	2.64	2.36	3.25	3.81	5.26

Table 9: Performance of CGT compared with GraphRNN (G-RNN) and GT-GAN on MUTAG and MOLBACE Datasets. Here, each number indicates the absolute mean error of graph attributes between the predicted graph and the target graph. Lower the value, better the performance.

	WordNet		Citeseer		Ogbn-Arxiv	
	GED	**MAD**	**GED**	**MAD**	**GED**	**MAD**
GT-GAN (Guo et al., 2023)	-	21.28±31.46	-	6.67±7.66	-	21.66±18.13
GenStat (Zahirnia et al., 2024)	3.82±0.07	4.13±0.03	5.62±0.00	5.34±0.01	5.58±0.00	5.51±0.03
EDGE (Chen et al., 2023)	4.75±0.21	3.90±0.03	5.67±0.58	4.95±0.10	6.49±0.47	5.55±0.05
CGT (CNN)	3.14±0.22	1.22±0.05	5.41±0.60	2.47±0.11	4.74±0.38	2.23±0.14
CGT (MLP)	**1.61±0.05**	**0.62±0.06**	**4.42±0.29**	**2.43±0.03**	**3.38±0.07**	**1.64±0.05**

Table 10: Overall performance across datasets. Average mean absolute difference (MAD ↓) is the average of absolute mean error in satisfying target attributes. Lower is better. Here, "–" in GED (↓) indicates missing values as there were no predicted graphs with less than 10 nodes to calculate GED.

	MUTAG		MOLBACE	
	GED	**MAD**	**GED**	**MAD**
GT-GAN (Guo et al., 2023)	–	14.89±15.32	–	3.47±1.56
GenStat (Zahirnia et al., 2024)	–	2.77±1.94	–	4.27±2.27
EDGE (Chen et al., 2023)	–	2.79±0.20	–	3.59±0.45
CGT (CNN)	–	1.67±0.13	–	4.51±1.91
CGT (MLP)	–	**1.27±0.60**	–	**3.15±0.17**

Table 11: Overall performance across datasets. Average mean absolute difference (MAD $\downarrow$) is the average of absolute mean error in satisfying target attributes. Lower is better. Here, "–" in GED ($\downarrow$) indicates missing values as there were no predicted graphs with less than 10 nodes to calculate GED.