Conditional Structure Generation through Graph Variational Generative Adversarial Nets

Carl Yang, Peiye Zhuang, Wenhan Shi, Alan Luu, Pan Li (corresponding: jiyang3@illinois.edu)

University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

Problem Formulation

Conditional Structure Generation: Given a set of graphs with semantic contexts, learn to generate graphs of meaningful structures with related contexts.

Motivations

1) Decompose massive networks into small subnetworks with clear structures and contexts
2) Map network structures and context semantics in embedding spaces
3) Flexibly generate network structures under given semantics

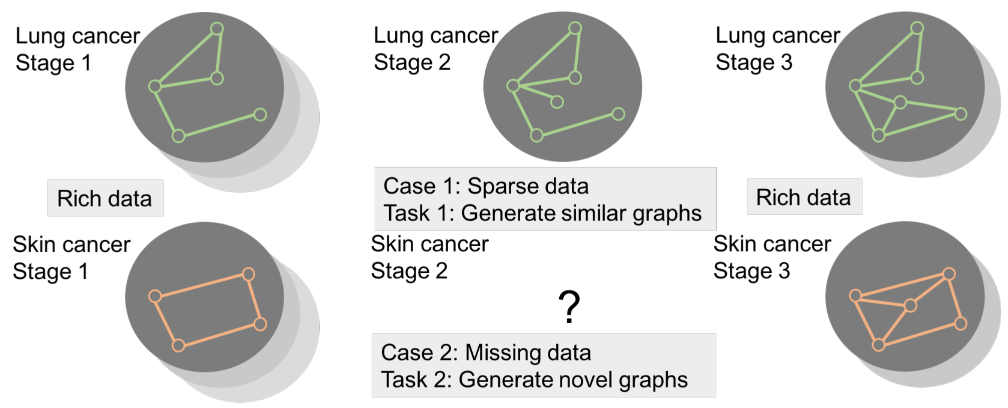
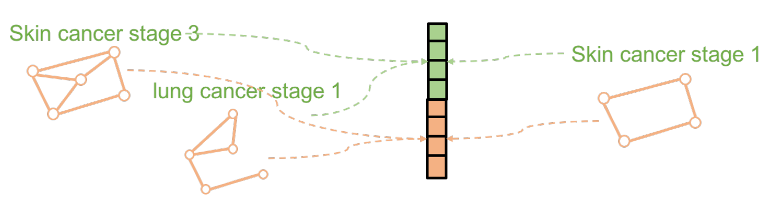


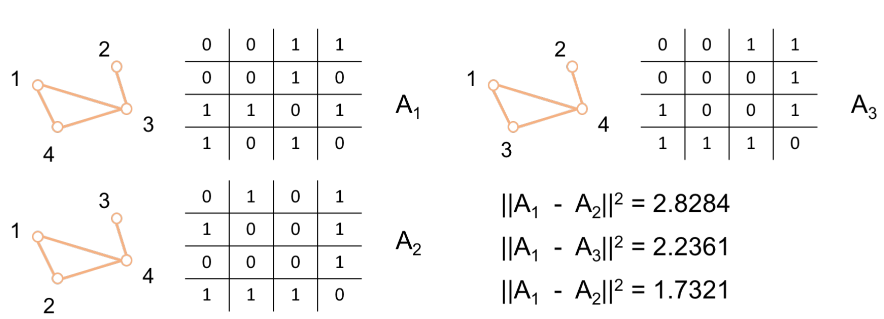
Figure: **Toy example of conditional structure generation:** Real-world networks nowadays are often associated with correlated semantic attributes/labels. This allows us to explore the possible correspondence between graph contexts and structures, which can be leveraged to generate structures for graphs with certain semantic contexts that are hardly observed.

Challenges and Requirements

Flexible context-structure conditioning: Learn a single representation from a set of graphs with variable sizes.



Permutation-invariant graph generation: Capture unique graph representations regardless of node ordering.



Technical Contributions

Contribution 1: A novel GCN-VAE framework with flexible conditioning function.

$$p(A|Z) = \prod_{i=1}^{n}\prod_{j=1}^{n} p(A_{ij}|\mathbf{z}_i, \mathbf{z}_j), \text{ with } p(A_{ij} = 1|\mathbf{z}_i, \mathbf{z}_j) = \sigma(\mathbf{f}(\mathbf{z}_i)^T \mathbf{f}(\mathbf{z}_j)), \quad (1)$$

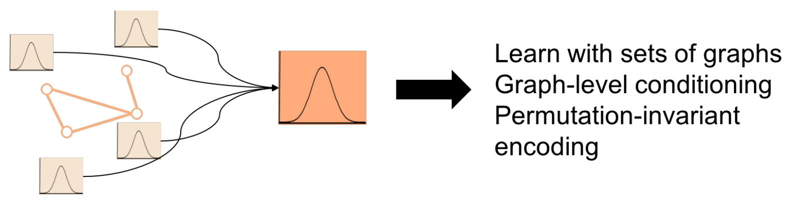


Figure: **Use a single distribution to jointly model all nodes.**

Contribution 2: A novel GCN-based graph discriminator to enable permutation invariance

$$\mathcal{L}_{gan} = \log(\mathcal{D}(A)) + \log(1 - \mathcal{D}(\mathcal{G}(Z_s))) + \log(1 - \mathcal{D}(\mathcal{G}(Z_c))). \quad (2)$$

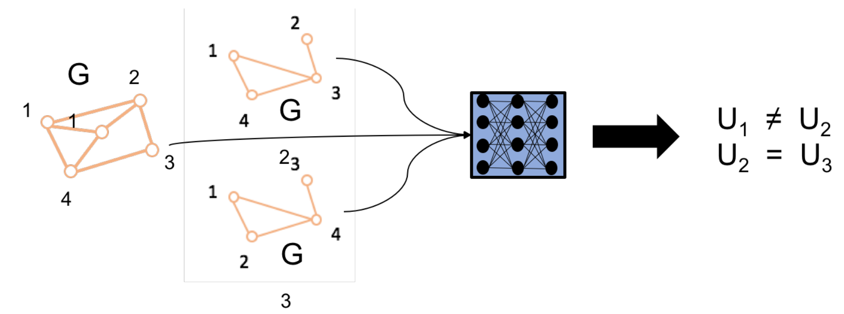


Figure: **End-to-end learnable graph structure representations.**

Overall Architecture of CondGen: GCN-VAE-GAN.

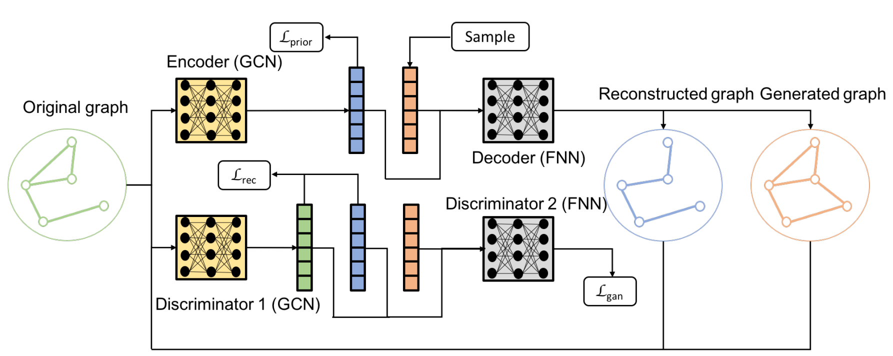


Figure: The upper part is a graph variational autoencoder, where we collapse the node embeddings into a single graph embedding, so as to enable flexible graph context-structure conditioning and allow training/generating of graphs with variable sizes. The lower part makes up for a graph generative adversarial nets, where we leverage GCN to guarantee permutation-invariant graph encoding, generation and comparison for reconstruction. Parameters in the decoder and generator networks as well as those in the two GCN networks in the encoder and discriminator are shared to further boost efficient and robust model inference.

Implementations: All code and data used in our experiments are publicly available at https://github.com/KelestZ/CondGen.

Experimental Evaluations

Datasets: We created two benchmark datasets, *i.e.*, a set of author citation networks from DBLP and a set of gene interaction networks from TCGA.
Baselines: We carefully adapt three state-of-the-art graph generation methods, *i.e.*, GVAE, NetGAN and GraphRNN, by concatenating the condition vectors to both the node features of the input graph and the output of the last encoding layer.
Protocols: We evaluate both tasks of mimicking similar seen graphs and creating novel unseen graphs, through visual inspection and graph property comparison (statistics we use include LCC (size of largest connected component), TC (triangle count), CPL (characteristic path length), MD (maximum node degree) and GINI (gini index), measuring different properties of graphs).

Graphs	Models	LCC	TC	CPL	MD	GINI
DBLP Seen	Real	96.00	48.54	3.696	11.62	0.3293
	GVAE	20.91**	21.76**	1.390*	2.32**	0.1964**
	NetGAN	21.15**	22.46**	1.641**	2.77**	**0.0568****
	GraphRNN	6.88*	69.32**	1.628**	7.06**	0.2446**
	CondGen(R)	6.70*	**7.70***	1.201*	**1.33**	0.1232*
	CondGen(S)	**6.00**	11.32	**0.963**	1.48	0.0959
DBLP Unseen	Real	102.50	58.21	4.982	14.29	0.3223
	GVAE	17.40**	17.02**	1.521**	3.53*	0.2479**
	NetGAN	29.57**	39.85**	1.494**	3.71**	**0.0812**
	GraphRNN	6.43	73.21**	1.305*	6.43**	0.1447**
	CondGen(R)	9.25*	10.50	1.445**	**1.92**	0.1418**
	CondGen(S)	**6.33**	**10.17**	**1.162**	**1.92**	0.0861
TCGA Seen	Real	177.34	8913.20	4.171	38.27	0.4192
	GVAE	54.82**	2396.94*	1.538	14.10**	0.2035**
	NetGAN	32.02**	3614.61**	1.702**	17.61**	0.1289*
	GraphRNN	**16.20***	2881.68**	1.899**	18.78**	0.2726**
	CondGen(R)	34.42**	2594.16**	1.542	9.50	0.1509**
	CondGen(S)	23.72	**2076.05**	**1.524**	**8.32**	**0.1093**
TCGA Unseen	Real	177.91	8053.18	4.143	34.34	0.4154
	GVAE	37.18**	2768.55**	**1.324***	13.03**	0.1497**
	NetGAN	31.36**	3557.91**	1.645*	18.45**	0.1277**
	GraphRNN	**15.73****	2605.73**	1.859**	13.55**	0.2647**
	CondGen(R)	27.77*	3083.81**	1.362*	10.86*	0.1413**
	CondGen(S)	23.97	**2058.95**	1.522	**8.68**	**0.1003**

Table: Performance evaluation over compared algorithms regarding several important graph statistical properties. The Real rows include the values of real graphs, while the rest are the *absolute values of differences* between graphs generated by each algorithm and the real graphs. Therefore, smaller values indicate higher similarities to the real graphs, thus better overall performance. We conduct paired t-test between each baseline and CondGen(S), scores with * and ** passed the significance tests with $p = 0.05$ and $p = 0.01$, respectively.

More results: For more experimental results on runtimes, visual inspections, and training details, please refer to our paper and supplementary materials at http://jiyang3.web.engr.illinois.edu/.