Group Meeting - October 23, 2020

Paper review & Research progress

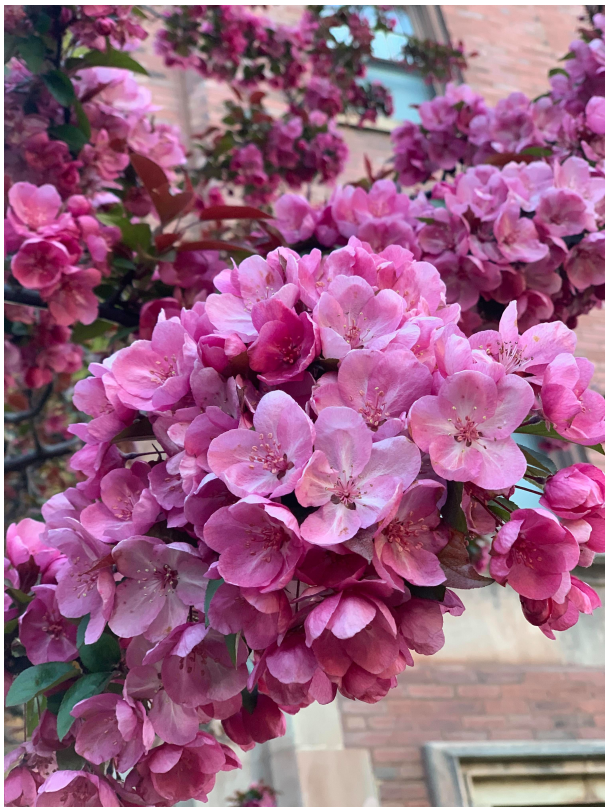
Truong Son Hy *

*Department of Computer Science
The University of Chicago

Ryerson Physical Lab

Lao Tzu

1. A journey of a thousand miles must begin with a single step.
2. Nature does not hurry, yet everything is accomplished.



My apartment, Spring 2020

Papers

1. **Dirichlet Graph Variational Autoencoder** (NeurIPS 2020)
   https://arxiv.org/abs/2010.04408
2. **Generative 3D Part Assembly via Dynamic Graph Learning** (NeurIPS 2020)
   https://arxiv.org/abs/2006.07793
3. **Deep imitation learning for molecular inverse problems** (NeurIPS 2019)
   https://papers.nips.cc/paper/8744-deep-imitation-learning-for-molecular-inverse-problems

Paper 3

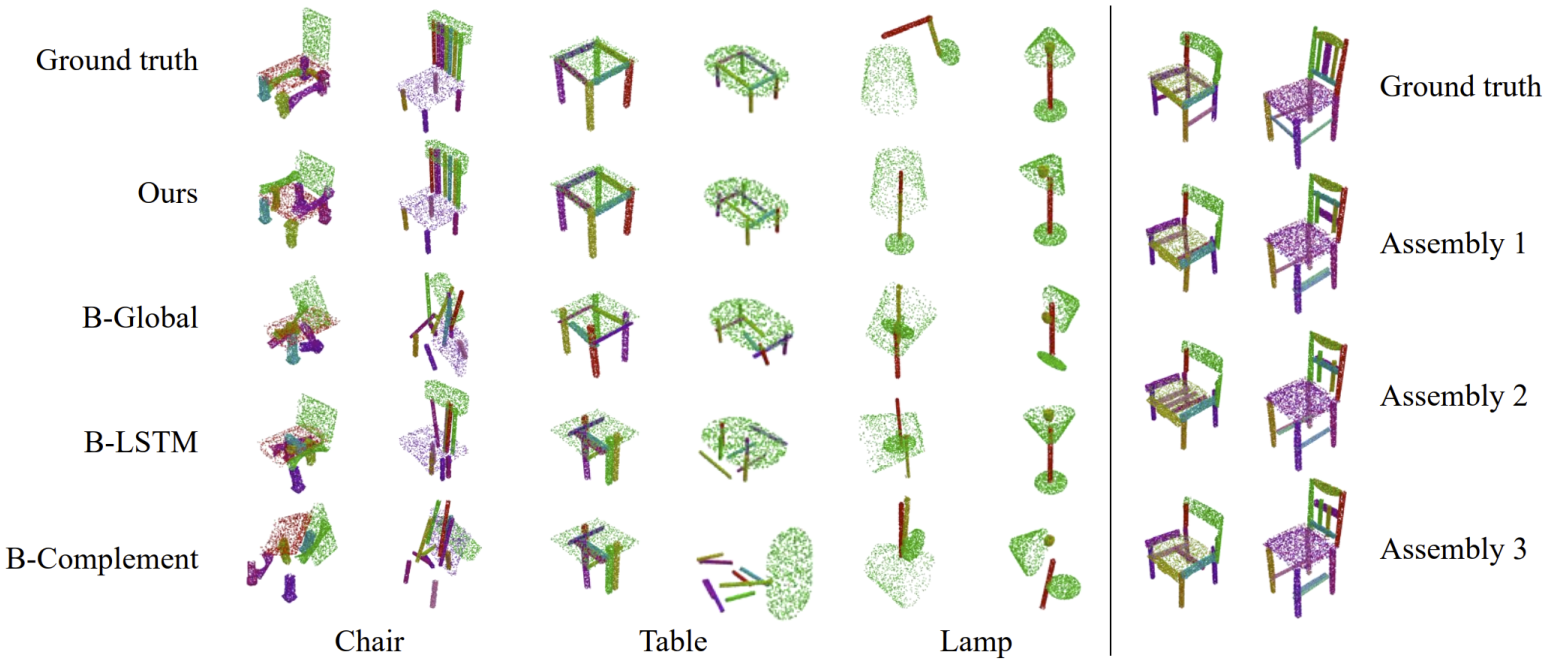
Generative 3D Part Assembly via Dynamic Graph Learning (NeurIPS 2020)

Jialei Huang, Guanqi Zhan, Qingnan Fan, Kaichun Mo, Lin Shao, Baoquan Chen, Leonidas Guibas, Hao Dong

`https://arxiv.org/abs/2006.07793`

Generative 3D Part Assembly via Dynamic Graph Learning (1)

3D part assembly:



To our research

Given the molecular structures as parts, how can we **assemble** them into a single valid molecule that has an expected property? In an equivariant way.

Generative 3D Part Assembly via Dynamic Graph Learning (2)

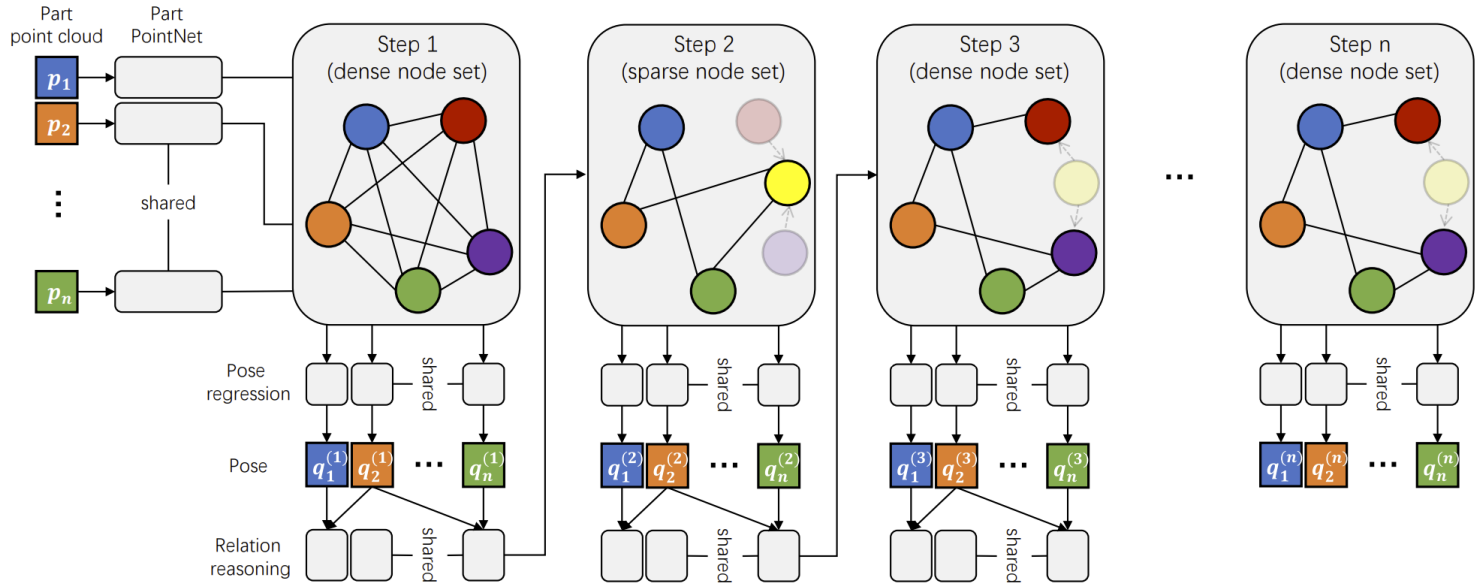


Figure 1: The proposed dynamic graph learning framework. The iterative graph neural network backbone takes a set of part point clouds as inputs and conducts 5 iterations of graph message-passing for coarse-to-fine part assembly refinements. The graph dynamics is encoded into two folds, (a) reasoning the part relation (graph structure) from the part pose estimation, which in turn also evolves from the updated part relations, and (b) alternatively updating the node set by aggregating all the geometrically-equivalent parts (the red and purple nodes), *e.g.* two chair arms, into a single node (the yellow node) to perform graph learning on a sparse node set for even time steps, and unpooling these nodes to the dense node set for odd time steps. Note the semi-transparent nodes and edges are not included in graph learning of certain time steps.

Paper 2

Deep imitation learning for molecular inverse problems (NeurIPS 2019)

Eric Jonas

https://papers.nips.cc/paper/8744-deep-imitation-learning-for-molecular-inverse-problems

Deep imitation learning for molecular inverse problems (1)

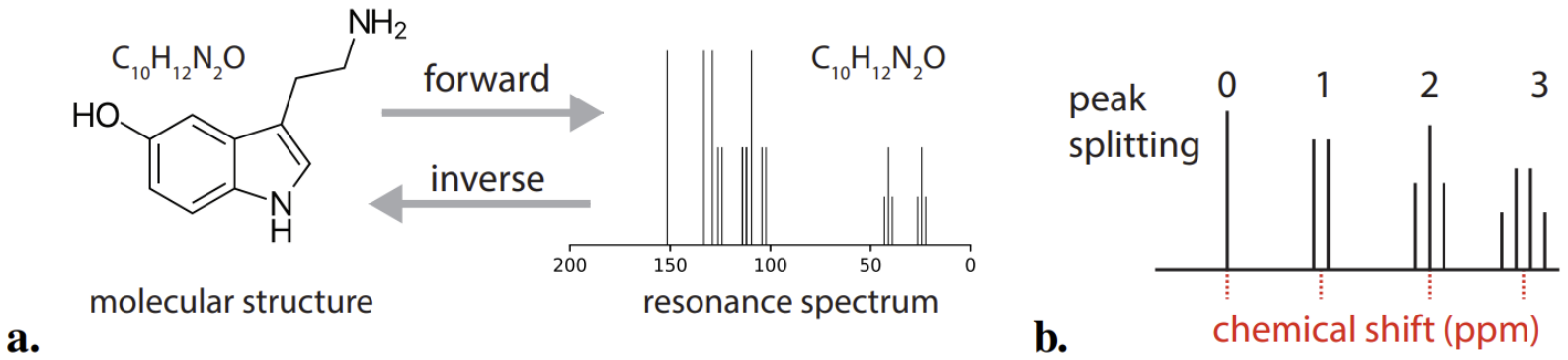


Figure 1: **a.** The **forward problem** is how to compute the spectrum of a molecule (right) given its structure (left). Spectroscopists seek to solve the corresponding **inverse problem**, working backward from spectra towards the generating structure. **b.** Various properties measured in a spectrum, including the chemical shift value and the degree to which each peak is split into multiplets.

Deep imitation learning for molecular inverse problems (2)

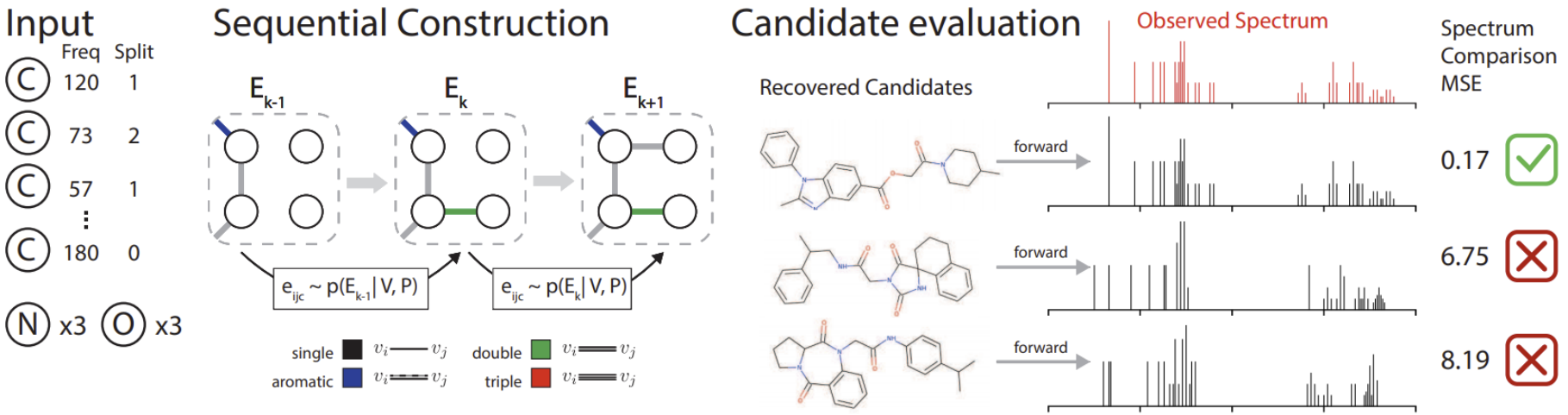


Figure 2: a.) Our input is a molecular formula indicating the number of atoms (vertices) of each element (color) along with per-vertex properties P measured for a subset of those vertices (in this case, carbon nuclei). b.) We sequentially construct a molecule by sampling the next edge (i, j) and edge label c conditioned on the existing edge set as well as vertices V and vertex properties P. c.) We end up with a sampled collection of candidate molecules, which we can then pass back through our forward model to compute their spectra, and validate against the true (observed) spectrum.

Deep imitation learning for molecular inverse problems (3)

layer of relational network

Input: $v_i^{(\text{in})}, e_{ij}^{(\text{in})}$

Output: $v_i^{(\text{out})}, e_{ij}^{(\text{out})}$

$$e_{ij} \longleftarrow L_e^k e_{ij}^{(\text{in})}$$
$$v_i' \longleftarrow L_v^k v_i^{(\text{in})}$$
$$e_{ij}' \longleftarrow \phi_e(e_{ij} + v_i' + v_j')$$
$$v_i^e \longleftarrow \max_i e_{ij}'$$
$$v_i^c \longleftarrow \phi_v(G(v_i^e, v_i^{(\text{in})}))$$
$$v_i^{(\text{out})} \longleftarrow v_i^c + v_i^{(\text{in})}$$
$$e_{ij}^{(\text{out})} \longleftarrow e_{ij}' + e_{ij}^{(\text{in})}$$

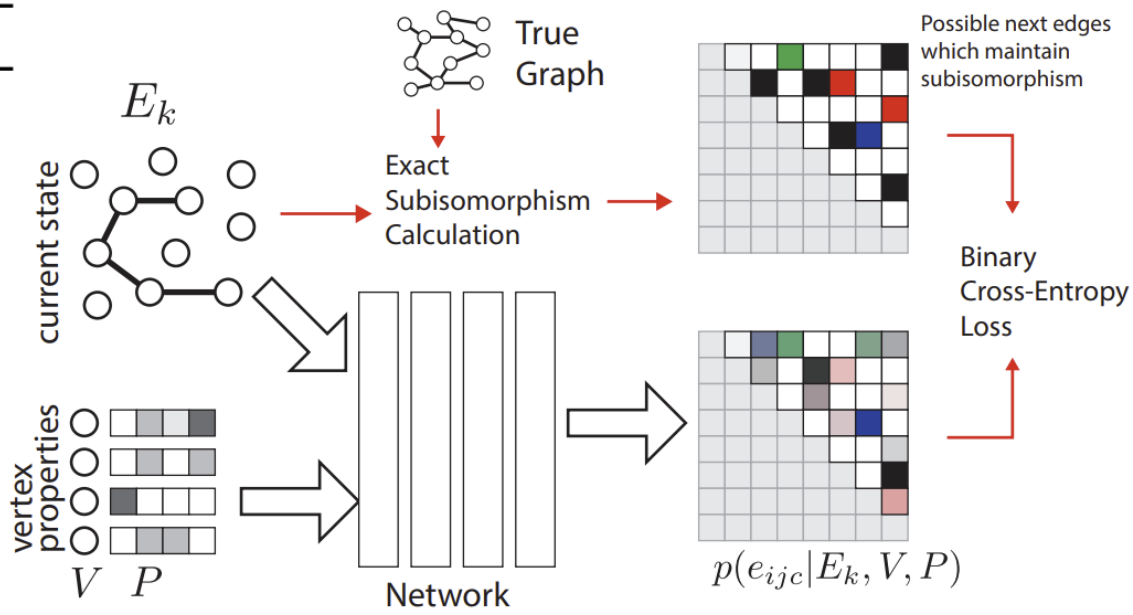


Figure 3: Each layer k of the network transforms per-vertex $v^{(\text{in})}$ and per-edge features $e^{(\text{in})}$ into per-vertex $v^{(\text{out})}$ and per-edge $e^{(\text{out})}$ output features. At train time we take true graphs, randomly delete a subset of edges, and exactly compute which single edges could be added back into the graph and maintain subisomorphism. We minimize the binary cross-entropy loss between the output of our network and this matrix of possible next edges.

Deep imitation learning for molecular inverse problems (4)

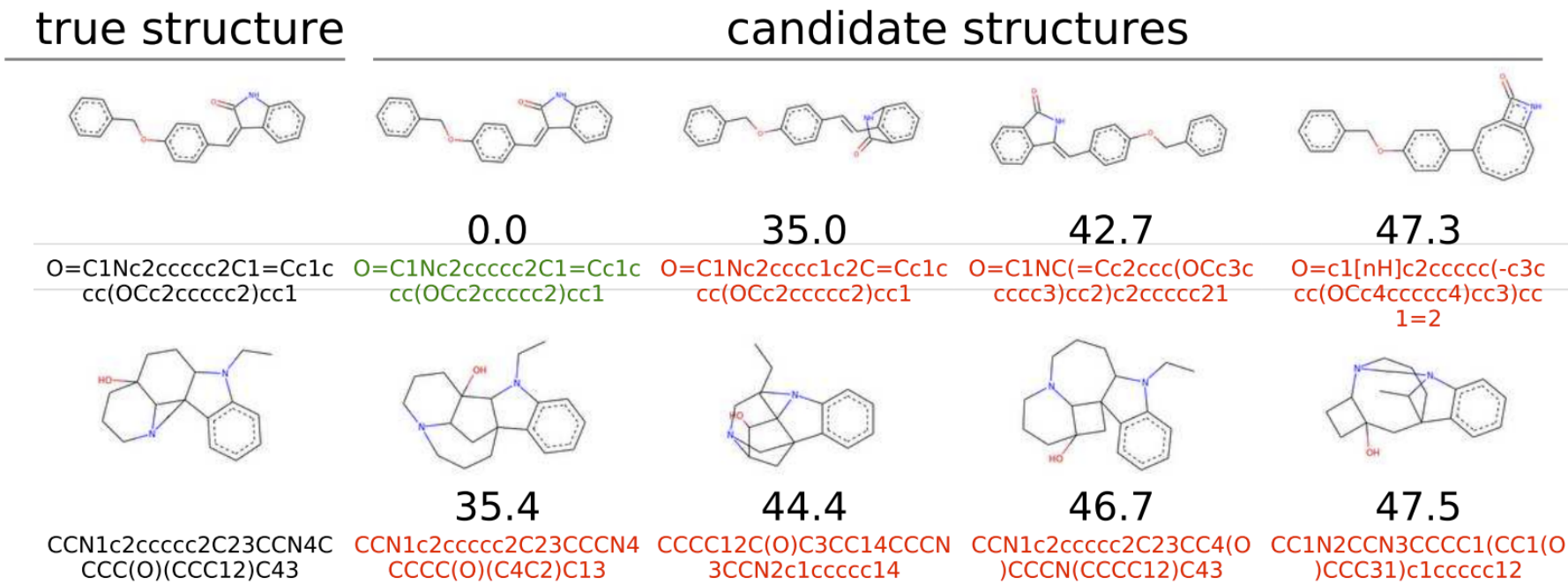


Figure 4: Example recovered structures. The left-most column is the true structure and associated SMILEs string, and the right are the candidate structures produced by our method, ranked in order of spectral reconstruction error (number below). Color indicates correct SMILEs string. The top row shows a molecule for which the correct structure was recovered, and the bottom row is an example of a failure to identify a structure.

Paper 1

Dirichlet Graph Variational Autoencoder (NeurIPS 2020)
Jia Li, Tomasyu Yu Jiajin Li, Honglei Zhang, Kangfei Zhao, YU Rong, Hong Cheng
https://arxiv.org/abs/2010.04408

Note: This work is **not** convincing to me. The idea is not new comparing to:

1. **Dirichlet Variational Autoencoder**, https://arxiv.org/abs/1901.02739
2. **Stick-Breaking Variational Autoencoders** (ICLR 2017), https://arxiv.org/abs/1605.06197

Proposals

Proposals

- **Main idea:** Replace Gaussian variables by the Dirichlet distributions in latent modeling of VAEs, such that the latent factors can be adopted to describe graph cluster memberships.
- **Claim:** Interpret the reconstruction term as a **balanced graph cut**.
- Propose a new variant of GNN named Heatts utilizing the Taylor series for fast computation of heat kernels.

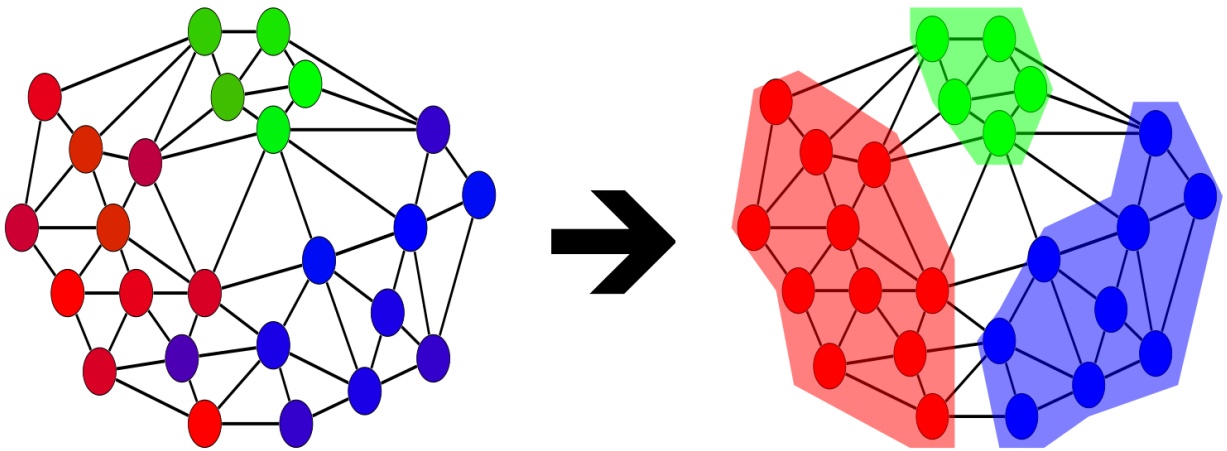
Background – Balanced graph cut (1)

Definition – Graph cut

Graph cut is defined as the number of edges between different clusters:

$$\frac{1}{K}\sum_{k}\text{cut}(V_k, \overline{V_k})$$

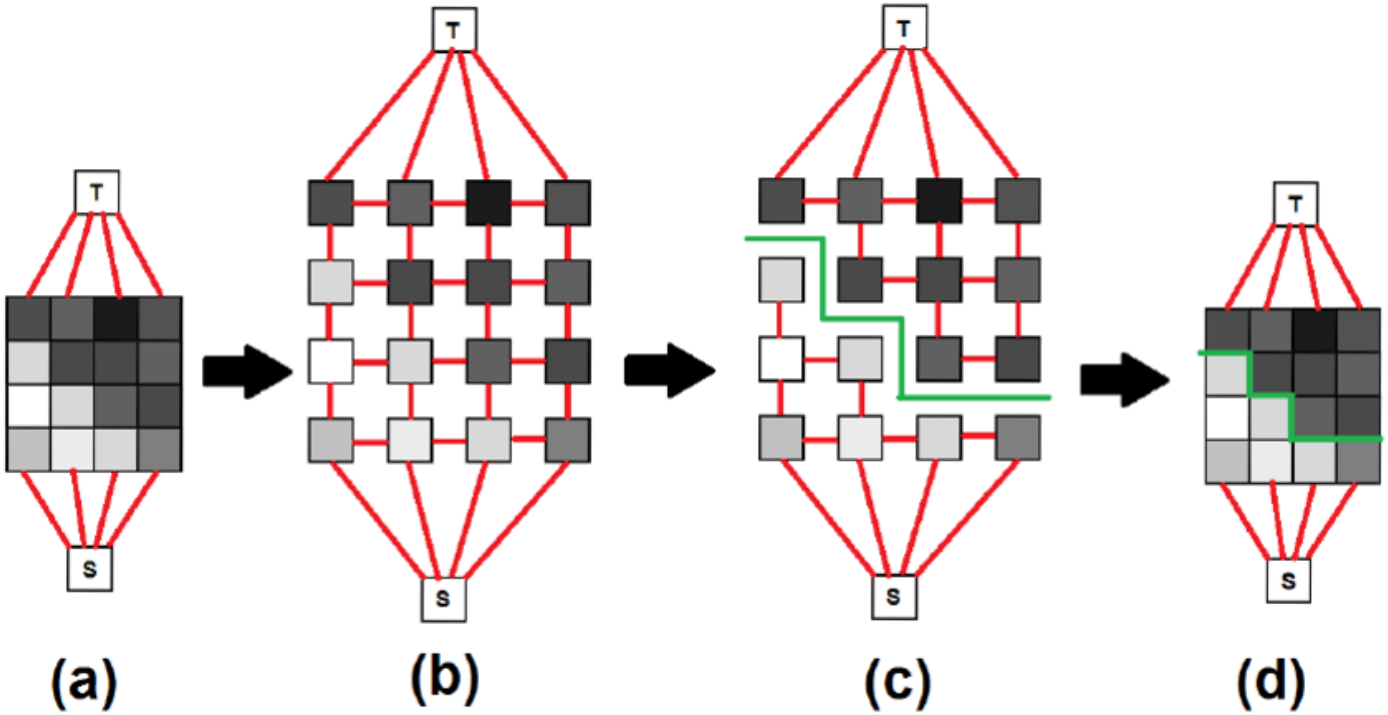
where V_k is the node set assigned to cluster k, $\overline{V_k} = V - V_k$, and $\text{cut}(V_k, \overline{V_k}) = \sum_{i \in V_k, j \in \overline{V_k}} A_{ij}$.



Source: `https://github.com/loicland/cut-pursuit`

Background – Balanced graph cut (2)

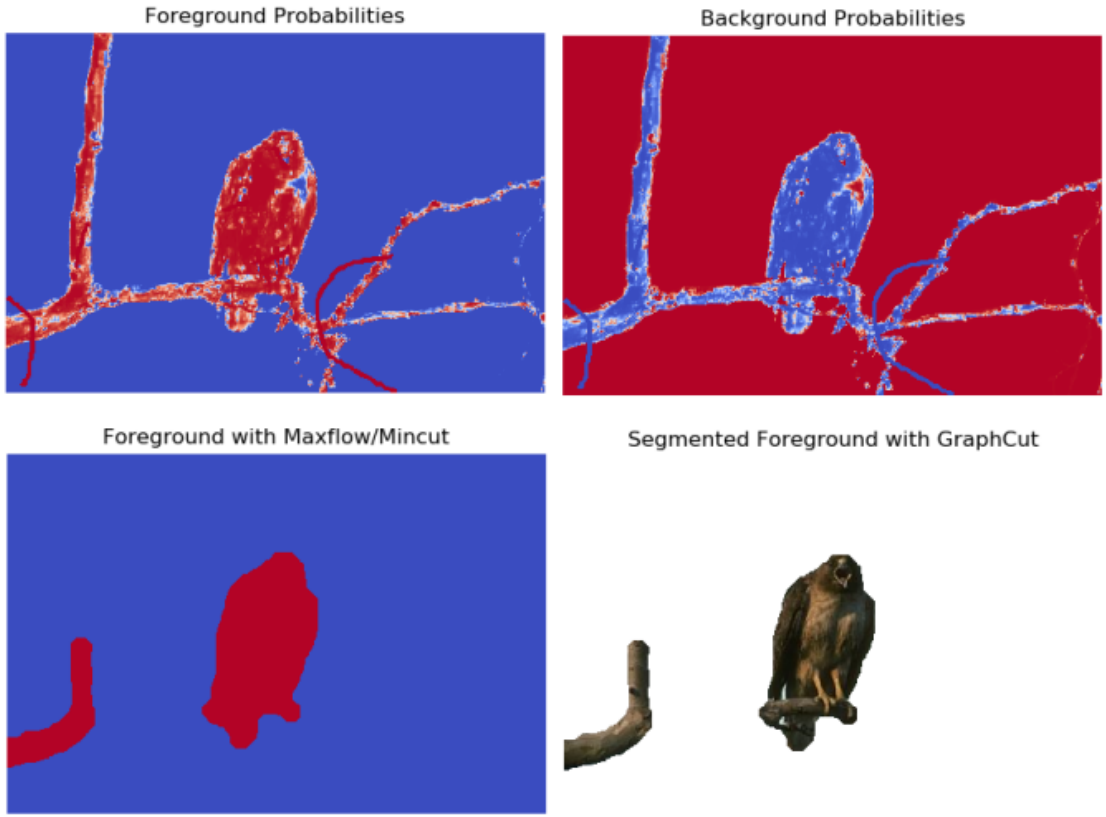
Using the max-flow/min-cut approach for image segmentation:



Source: *Tracing liquid level and material boundaries in transparent vessels using the graph cut computer vision approach*, Sagi Eppel

Background – Balanced graph cut (3)

Markov Random Field:



Source: https://sandipanweb.wordpress.com/2018/02/11/interactive-image-segmentation-with-graph-cut/

Background – Balanced graph cut (4)

Definition – Graph cut (alternative)

$$\frac{1}{K}\sum_{k}(C_{:,k}^{T}DC_{:,k} - C_{:,k}^{T}AC_{:,k}) = \frac{1}{K}\text{trace}(C^{T}LC)$$

where $C \in \{0,1\}^{N\times K}$ is the cluster indicator, $C_{ik} = 1$ represents node i belongs to cluster k, and 0 otherwise. D denotes the degree matrix, and $L = D - A$ is the un-normalized graph Laplacian.

- $C_{:,k}^{T}DC_{:,k}$ = number of edges with at least 1 end point in V_k
- $C_{:,k}^{T}AC_{:,k}$ = number of edges within cluster V_k

Background – Balanced graph cut (5)

Graph cut favors **imbalanced** clustering $\rightarrow$ utilize **ratio cut**.

Definition – Ratio cut

$$\frac{1}{K}\sum_{k}\frac{C_{:,k}^{T}LC_{:,k}}{|V_k|}$$

where $|V_k| = C_{:,k}^{T}C_{:,k}$ counts the number of nodes within cluster k.

The minimum of the function $\sum_{k=1}^{K}\frac{1}{|V_k|}$ is achieved if all $|V_k|$ are the same. Ratio cut is an NP-hard problem $\rightarrow$ spectral clustering.

Background – Balanced graph cut (6)

Spectral clustering:

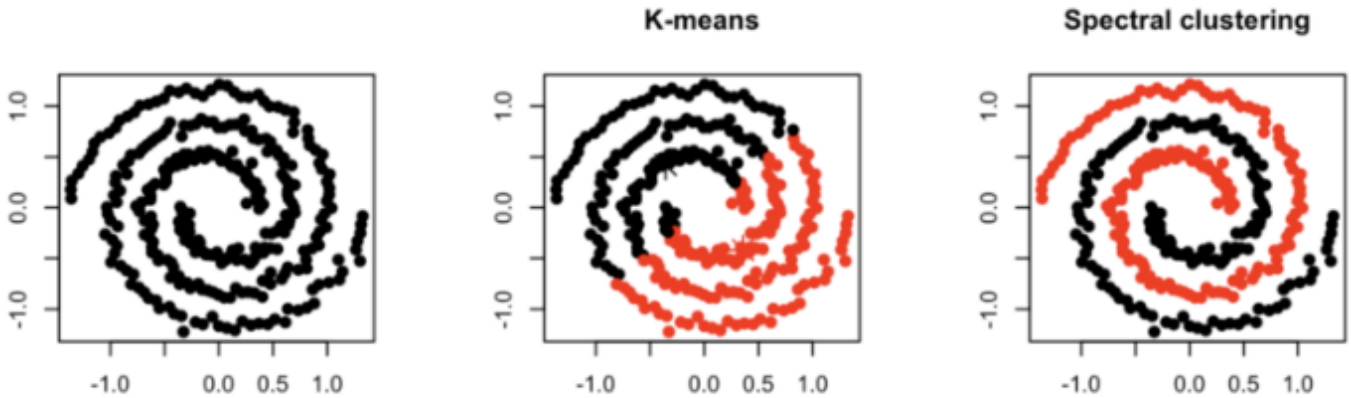
1. Compute the normalized graph Laplacian: $L = D^{-1/2}LD^{-1/2}$.
2. For k clusters, compute the first (smallest) k eigenvectors $(v_1, .., v_k)$ associating with k eigenvalues $(\lambda_1, .., \lambda_k)$.
3. Stack the eigenvectors vertically (column-by-column). Represent each node by a row of this new matrix.
4. Perform K-Mean to cluster these nodes into clusters V_1, .., V_k.

Background – Balanced graph cut (7)

Spectral clutering: Smallest K eigenvalues $\rightarrow$ **low-pass filter** $g_{\text{id}}(\lambda_i)$ with threshold λ_K defined as

$$g_{\text{id}}(\lambda_i) = \mathbb{I}[\lambda_i \leq \lambda_K]$$

The difference between the 2 can easily be shown by this illustration:



Source: `https://medium.com/@SeoJaeDuk/archived-post-spectral-clustering-45c478ee0e30`

GNNs and VAEs

1. **Encoder:** a variational posterior $q_\phi(Z|G)$ parameterized by GNNs, with a prior $p(Z)$ acting as a regularization for $q_\phi(Z|G)$.
2. **Decoder:** a generative distribution $p_\theta(A|Z)$.

Maximnizing Evidence Lower Bound (ELBO):

$$\mathcal{L}_{\text{ELBO}}(\phi, \theta; G) = -\text{KL}(q_\phi(Z|G)||p(Z)) + \mathbb{E}_{q_\phi(Z|G)} \log p_\theta(A|Z)$$

Simplifying Graph Convolutional Networks (ICML 2019) [Wu et al., 2019] `https://arxiv.org/abs/1902.07153`:
GCN is a linear spectral filter with $g_c(\lambda_i) = 1 - \lambda_i \rightarrow$ deviates the low-pass characteristics.

Dirichlet Graph Variational Autoencoder (1)

Main idea: Replace Gaussian variables by the Dirichlet distributions in latent modeling of VAEs, such that the latent factors can be adopted to describe graph cluster memberships.

$$q_\phi(Z|A,X) = \prod_{i=1}^{N} q_{\phi_i}(z_i|A,X)$$

is the variational family in which the variational marginals $q_{\phi_i}(z_i|A,X)$ are assumed to follow the Dirichlet distributions.

Laplace approximation: to approximate the Dirichlet distributions with the logistic normal distribution.

Dirichlet Graph Variational Autoencoder (2)

The parameters for the variational marginals $q_{\phi_i}(z_i|A, X)$ are specified by a multi-layer GNN:

$$\mu^0, \sigma^0 = GNN_\phi(A, X)$$

We can approximate the Dirichlet distributions $q_\phi(z_i|G)$ by sampling $\epsilon \sim \mathcal{N}(0, 1)$ and compute

$$z_i = \text{softmax}(\mu^0 + (\Sigma^0)^{1/2}\epsilon)$$

where $\Sigma^0 = \text{diag}(\sigma^0) \in \mathbb{R}^{K \times K}$.

$Z = \{z_i\}_{i=1}^N$ coincides with the **relaxed** graph cluster memberships C like in spectral clustering.

Dirichlet Graph Variational Autoencoder (3)

We rewrite the prior $p(z_i) = \text{Dir}(\alpha)$ as the logistic normal distribution with mean μ^1 and covariance matrix Σ^1:

$$\mu_k^1 = \log \alpha_k - \frac{1}{K} \sum_i \log \alpha_i$$

$$\Sigma_{kk}^1 = \frac{1}{\alpha_k}\left(1 - \frac{2}{K}\right) + \frac{1}{K^2} \sum_i \frac{1}{\alpha_i}$$

The KL divergence between two logistic normal distributions as:

$$\mathcal{D}_{\text{KL}}(q_\phi(z_i|G)||p(z_i)) = \frac{1}{2}\Big\{\text{trace}((\Sigma^1)^{-1}\Sigma^0) +$$

$$+(\mu^1 - \mu^0)^T(\Sigma^1)^{-1}(\mu^1 - \mu^0) - K + \log\frac{|\Sigma^1|}{|\Sigma^0|}\Big\}$$

Dirichlet Graph Variational Autoencoder (4)

Decoder:

$$p(A|Z) \propto \prod_{A_{ij}=1} \exp f(C_i, C_j) \prod_{A_{ij}=0} \exp(1 - f(C_i, C_j))$$

where f denotes a distance metric:

1. $f(C_i, C_j) = C_i^T C_j$
2. $f(C_i, C_j) = 1 - \text{MSE}(C_i, C_j)$ where MSE = mean squared error.

Reconstruction term as balanced graph cut

Claim

Maximizing the reconstruction term of DGVAE is equivalent to minimizing the spectral relaxed graph cut and a regularization that encourages balanced cluster size.

Note

The claim is **trivial**!

The soft membership (assignment) $C_i \in \Delta_{K-1}$ where Δ_{K-1} is the $(K-1)$-simplex. We can regard this row C_i as an independent sample drawn from the posterior Dirichlet distributions.

Taylor series approximation for heat kernels (1)

Motivation

We need a low-pass graph filter that:

1. retains the low eigenvectors, and drops the high eigenvectors of the graph Laplacian L.
2. is fast, not involving the explicit eigendecomposition of L – computationally expensive.

Consider the heat kernel $g_s(\lambda) = e^{-s\lambda}$ where $s > 0$ is a scaling hyper-parameter. The spectral graph convolutions on a signal $x \in \mathbb{R}^N$:

$$g_s * x = U\text{diag}(g_s(\lambda_1), .., g_s(\lambda_N))U^T x = Ug_s(\Lambda)U^T x$$

where U is the eigenvector matrix of the normalized graph Laplacian $L = U\Lambda U^T$.

Taylor series approximation for heat kernels (2)

We apply the Taylor series approximation on $g_s(\Lambda)$:

$$g_s * x = U \sum_n \frac{(-1)^n}{n!} s^n \Lambda^n U^T x = \sum_n \frac{(-1)^n}{n!} s^n L^n x$$

Heatts – Taylor approximation:

- Message function:

$$M^\ell = \sum_{n=0}^{3} \frac{(-1)^n}{n!} s^n L^n H^\ell$$

- Vertex update function:

$$H^{\ell+1} = \mathrm{ReLU}(M^\ell W)$$

where $H^0 = X$. W is the parameter set of be learned.

Experiments (1)

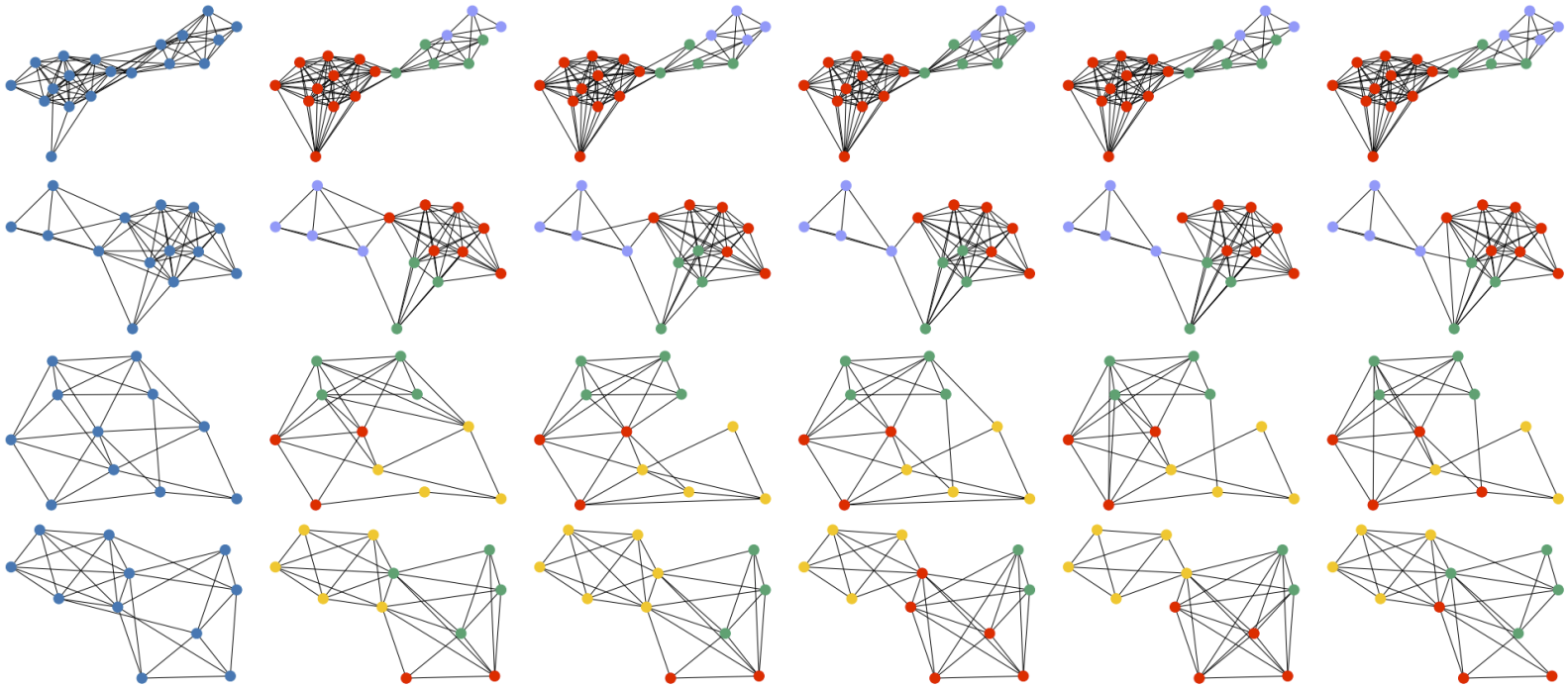


Figure 2: Left one in blue: the input graphs. Right five in colors: graph samples generated by DGVAE, where colors indicate latent cluster memberships with $K = 3$.

Experiments (2)

Table 1: Test graph generation comparison of different methods

	Erdos-Renyi		Ego		Regular		Geometric		Power Law		Barabasi-Albert	
	NLL	RMSE	NLL	RMSE	NLL	RMSE	NLL	RMSE	NLL	RMSE	NLL	RMSE
GAE	0.647	0.535	0.356	0.346	0.523	0.455	0.583	0.370	0.580	0.401	0.553	0.428
VGAE	1.010	0.609	0.917	0.492	0.914	0.452	0.813	0.524	0.901	0.485	0.894	0.501
Graphite-AE	0.678	0.529	0.370	0.333	0.526	0.395	0.851	0.385	0.541	0.399	0.557	0.390
Graphite-VAE	1.087	0.602	0.896	0.496	0.983	0.474	0.846	0.536	0.938	0.466	0.925	0.482
Abl-AE	0.646	0.530	0.349	0.400	0.497	0.429	0.472	0.350	0.536	0.419	0.536	0.383
Abl-VAE	0.760	0.512	0.541	0.445	0.601	0.454	0.682	0.475	0.638	0.395	0.678	0.430
DGAE	**0.239**	**0.186**	**0.250**	**0.231**	**0.305**	**0.282**	**0.406**	**0.182**	**0.383**	0.415	**0.308**	0.214
DGVAE	0.286	0.249	0.436	0.274	0.516	0.340	0.537	0.233	0.519	**0.255**	0.346	**0.194**

NLL = Negative log-likelihood

RMSE = Root mean square error

Note: For the Dirichlet one, AE seems to be better than VAE.

Experiments (3)

Table 2: Cluster performance comparison of different methods

	Pubmed			Citeseer			Wiki		
	ACC (%)	NMI (%)	F1 (%)	ACC (%)	NMI (%)	F1 (%)	ACC (%)	NMI (%)	F1 (%)
SC	58.3 ± 0.5	19.0 ± 0.6	43.2 ± 0.4	23.9 ± 1.4	5.9 ± 3.5	29.5 ± 2.6	23.6 ± 3.7	19.3 ± 3.2	17.3 ± 2.5
N2v&K	67.7 ±1.2	**29.5 ±1.3**	66.3 ±1.1	41.3 ±1.1	16.7 ±0.8	39.5 ±1.3	34.9 ± 1.8	31.1 ± 2.2	**30.3 ±1.3**
GAE&K	64.2 ±1.9	24.0 ±1.5	64.4 ±1.1	41.2 ±0.9	20.8 ±1.2	40.1 ±1.2	25.1 ± 1.9	26.7 ± 1.7	19.8 ± 1.8
VGAE&K	62.0 ± 3.0	20.4 ± 1.7	62.5 ± 2.8	43.4 ±3.3	22.7 ±0.9	41.8 ±3.0	32.2 ± 2.1	30.2 ± 2.8	29.3 ± 2.2
Abl-AE	66.3 ±1.4	25.7 ±1.9	66.2 ±1.7	46.1 ±1.9	24.3 ±2.2	40.1 ±2.2	38.1 ±2.3	33.8 ±1.6	24.7 ±2.1
Abl-VAE	62.6 ±2.1	24.2 ±2.7	61.4 ±2.5	40.2 ±2.7	16.1 ±2.2	38.5 ±2.4	36.2 ±2.3	31.4 ±1.4	25.9 ±2.7
DGAE	**68.4 ±1.9**	28.8 ±2.1	**67.3 ±2.3**	**51.3 ±2.1**	**27.2 ±1.5**	**49.4 ±1.8**	**38.9 ±1.8**	**36.9 ±1.5**	27.7 ± 2.3
DGVAE	64.9 ±2.0	25.8 ±2.5	66.5 ±2.2	44.9 ±2.8	19.4 ±2.7	41.9 ±3.1	37.5 ±3.3	31.7 ±2.6	28.7 ±1.9

NMI = Normalized Mutual Information

Note: For the Dirichlet one, AE seems to be better than VAE.

Implementation

`https://github.com/HyTruongSon/LibCCNs/tree/master/pytorch/small_graphs`

Note:

- Seems to have a good convergence.
- The KL divergence loss (from Laplace approximation) seems to be **numerically unstable**.