Deep Node Clustering based on Mutual Information Maximization

Soheila Molaei[a,*], Nima Ghanbari Bousejin[a,*], Hadi Zare[a,**], Mahdi Jalili[b]

[a] *Department of Network Science and Technologies, University of Tehran*
[b] *School of Engineering, RMIT University*

Abstract

Variational Graph Autoencoders (VGAs) are generative models for unsupervised learning of node representations within graph data. While VGAs have been achieved state-of-the-art results for different predictive tasks on graph-structured data, they are susceptible to the over-pruning problem where only a small subset of the stochastic latent units are active. This can limit their modeling capacity and their ability to learn meaningful representations. In this paper, we present SOLI (Stacked auto-encoder for nOde cLusterIng), an information maximization approach for learning graph representations by leveraging maximal cliques. SOLI relies on aggregating useful representations by assigning clique-based weights to various edges in a neighborhood while maximizing mutual information. The learned representations are mindful of graph patches centered around each node, and can be used for a range of downstream tasks, and thus encouraging more active units. We demonstrate strong performance across three graph benchmark datasets.[1]

Keywords: Graph representation learning, Graph neural network, Graph auto-encoder, Node clustering, Link prediction

1. Introduction

In recent years, numerous attempts were made to extend neural networks to graph data structures such as social networks [1, 2], recommender graphs

*Equal Contribution
**Corresponding author
[1] Code is available at https://github.com/SoheilaMolaei/SOLI.

[3], or molecular graphs [4]. Graph networks have recently made significant strides on node classification [5, 6, 7], link prediction [8], and clustering [9, 10]. However, high dimensional data and computational complexity have made such graph-based computational tasks deeply challenging [11].

A strong and increasingly common approach for dealing with complex graph data is node embedding. By working in the embedding space, established learning methods can be used and the complexity of integrating complex node interactions can be bypassed [12]. Significant progress has recently been made, notably with graph convolutional network (GCN) [5] and variational graph auto-encoder (VGAE) [9] in order to learn useful node embeddings.

While powerful, variational graph auto-encoders are susceptible to the over-pruning problem as they are unable to obtain enough data about the input graph and their latent variables generate standard Gaussian noise [13]. This greatly mitigates the modeling capacity of variational graph auto-encoders to learn meaningful latent representations. Quite a few methods have been recommended to overcome over-pruning by training schemes. VGAE [9] enforces minimum KL contribution from subsets of latent units while [14] uses KL cost annealing. Although these schemes mitigate the over-pruning problem, they are hand-tuned and take away the principled regularization scheme that is built into VAE.

In this paper, we propose an unsupervised node representation learning pipeline that is based upon maximal clique and mutual information. SOLI relies on aggregating node representations by focusing on the most significant parts of the input graph, while maximizes the local mutual information. These representations are then retrieved and used by a graph auto-encoder (GAE) with random walk-based regularization enforcing the latent representations to capture the graphs' local structural properties. We demonstrate that the learned patch representations are competitive on three standard datasets and outperform the state-of-the-art benchmarks in our experiments. Our contributions, which will be presented in the rest of the paper, are as follow:

- We introduce SOLI-GAE which learns a latent variable model by maximizing information across maximal cliques.
- We propose SOLI-VGAE that achieves high generative ability while mitigates the over-pruning problem by contrasting between global and local parts of a graph simultaneously.

- We demonstrate strong results on a range of link prediction and node clustering benchmark tasks.

2. Related Works

Several attempts have been made to extend neural networks to deal with graph-structured data. Our proposed approach builds upon a rich line of research on graph auto-encoders and graph embedding methods.

Graph auto-encoders [9, 15, 16] extend auto-encoders to graph-structured data. Their goal is to obtain node embeddings that preserve important characteristics of the original input. Variational graph auto-encoders are an unsupervised learning approach to graph data structures based on the variational auto-encoder [17]. The idea behind variational auto-encoders is that the encoder maps an input to a distribution instead of a fixed vector and feeds a sample from this distribution to the decoder. VGAE [9], naturally incorporates structural and contextual information of the graph using a GCN encoder and a simple inner product decoder. ARGA [18] uses convolutional auto-encoder to regularize the latent information to match a prior distribution, while RWR-GAE [19], proposes a method based on a random walk that regularizes the encoders' learned representations.

Graph embedding [20, 21] is a low-dimensional representation of graph-structured data. The Skip-gram model [22] is used by random-walk-based approaches such as DeepWalk [1] and node2vec [23] by treating walks as sentences while generating shortened random walks on graphs. Factorization based methods like GraRep [24] and NetMF [25] apply matrix factorization techniques to generate the embedding vectors. Deep learning-based approaches like Deep Graph Infomax (DGI) [26] and VGAE [9] use contrastive methods to learn graph representations. Although random walk-based approaches have been successfully used for unsupervised graph representation learning, the dominant algorithms are mutual information maximization in a self-supervised manner. Despite their effectiveness in many applications, random walk-based objectives emphasize maintaining the local graph structure, while ignoring global information, such as community structure.

SOLI aims to solve over-pruning by leveraging information maximization across maximal cliques. We seek to obtain node representations from the most relevant parts of the graph while capturing global representations of the entire graph, but, previous approaches [9, 18, 19] rely heavily on local parts of the input graph. Our graph auto-encoder utilizes these learned

representations for clustering by jointly optimizing cluster label assignment and learning features that are suitable for clustering with local structure preservation.

3. Preliminaries

Definition 1. Given a graph $G = (V, E)$ with two finite sets of V nodes and E edges, a clique is a subset of nodes such that each node in the set is adjacent to all other nodes in the set, i.e., a subset $C \subseteq V$ is a clique if and only if $(v, v') \in E$ for all nodes v and $v' \in C$ [27]. Provided the notion of cliques, a maximal clique in a graph is defined as follows:

Definition 2. A maximal clique in a graph is a clique that cannot be extended by adding one more neighboring node without compromising its connectivity property, i.e., a maximal clique is a clique which is not a subgraph of a larger clique. A clique with the largest number of nodes is the maximum clique.

4. Proposed Method

Motivation. We are interested in high-quality message passing by specifying clique-based edge weights, where nodes are able to attend to the most relevant parts of their neighborhood. Our approach relies on a mutual information maximization principle to learn representations, followed by a GAE which makes the learned embeddings exhibit discernible clustering.

Notation. We are given a graph G with N nodes represented as a pair (X, A) where $X \in \mathbb{R}^{N \times F}$ is the node feature matrix with F features per node, and $A \in \mathbb{R}^{N \times N}$ is the adjacency matrix. We further introduce a latent variable summarized in a D-dimensional vector representation, $Z \in \mathbb{R}^{N \times D}$. Additionally, our method assumes unweighted and undirected graphs, i.e. $A_{ij} = 0$ if there is no edge between i and j and $A_{ij} = 1$ otherwise.

Model. As an initial step, we utilize a clique function, $\mathcal{Q} : \mathbb{R}^{N \times N} \rightarrow \mathbb{R}^{N \times N}$, to find all maximal cliques from the original graph using Bron-Kerbosch algorithm [28] and sample one-hop neighboring nodes such that $A'_{ij} = c$ represents nodes i and j are in a $(c + 1)$-clique as shown in Figure 1. In extremely sparse settings, we find the two-hop neighboring nodes in addition to the one-hop neighborhood. Our proposed method allows to assign node-specific weights within a neighborhood. Indeed, densely connected nodes receive higher weights than those with sparse connections.

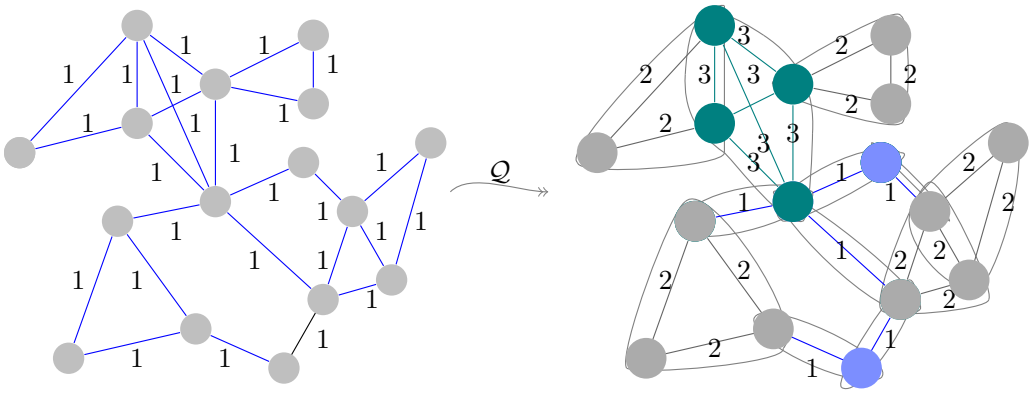


Figure 1: Clique function, $\mathcal{Q}$, applied to the input graph (**left**) to obtain the weighted graph (**right**), i.e. $A'_{ij} = c > 0$ if i and j are in a $(c+1)$-clique and $A'_{ij} = 0$ otherwise.

Our approach for unsupervised learning of node representations is to train an encoder to be contrastive by increasing the score on the original input (positive examples) and decreasing the score on the corrupted input (negative examples). We pass node features, X, and the weighted adjacency matrix, A', as our positive examples and three corrupted versions of the original graph as our negative examples through the encoder function, $\mathcal{E} : \mathbb{R}^{N\times N} \times \mathbb{R}^{N\times F} \rightarrow \mathbb{R}^{N\times\mathbb{F}'}$, thus $\mathcal{E}(X, A') = \{\vec{h_1}, \vec{h_2}, \ldots, \vec{h_N}\}$ shows patch representations. We find that expanding the pool of negative examples encourages diversity in them and makes the model more robust against fake inputs at test time. Following the intuitions from DGI [26], negative samples are provided from the input graph by using a corruption function, $\mathcal{C} : \mathbb{R}^{N\times N} \times \mathbb{R}^{N\times F} \rightarrow \mathbb{R}^{M\times M} \times \mathbb{R}^{M\times F}$, that maintains the adjacency matrix ($\tilde{A}' = A'$), but row-wise shuffles the features, ($\tilde{X} \neq X$) as it is designed to encourage the representations to properly encode structural similarities of different nodes in the graph; N and M denote the number of nodes in the original and alternative graphs, respectively. Our encoder is a one layer GCN model [5] with the following layer-wise propagation rule:

$$H^{(l+1)} = ReLU\left(\hat{D}^{\frac{-1}{2}}\hat{A}\hat{D}^{\frac{-1}{2}}H^{(l)}W^{(l)}\right) \tag{1}$$

where $\hat{A} = A' + I_N$ is the newly generated adjacency matrix added by an $N \times N$ identity matrix, I_N, and $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$ is its corresponding degree

matrix. $H^{(l)} \in \mathbb{R}^{N \times F'}$ and $H^{(l+1)} \in \mathbb{R}^{N \times F''}$ are the input and output feature matrices of l-th layer, and $W^{(l)} \in \mathbb{R}^{F' \times F''}$ is a layer-wise linear transformation applied to each node to compute $F'' = 16$ truncated features.

The discriminator, $\mathcal{D} : \mathbb{R}^F \times \mathbb{R}^F \rightarrow \mathbb{R}$, receives patch-summary pair, $(\vec{h_i}, \vec{s})$ and assigns a probability score by employing a bilinear scoring function, $\sigma(\vec{h_i}^T W \vec{s})$, to this pair corresponding to whether a given patch contains within the summary [26]; where σ is the logistic sigmoid nonlinearity and W is a learnable scoring matrix. Graph-level summary vector, $\vec{s}$, is obtained using a readout function, $\mathcal{R} : \mathbb{R}^{N \times F} \rightarrow \mathbb{R}^F$, which takes average of nodal features, $\sigma(\frac{1}{N} \sum_{i=1}^{N} \vec{h_i})$, with σ as the logistic sigmoid nonlinearity. We pair patch representations from the average of the alternative graphs with summary of the original graph for negative samples. Our objective follows the intuition from Deep InfoMax [29] that minimizes the binary cross-entropy loss between positive and negative samples:

$$\mathcal{L} = \sum_{i=1}^{N} \log \mathcal{D}\left(\vec{h_i}, \vec{s}\right) + \sum_{j=1}^{M} \log\left(1 - \mathcal{D}\left(\vec{\tilde{h}}_j, \vec{s}\right)\right) \tag{2}$$

We pass high-level node representations obtained from the most relevant parts of the graph through either a GAE or a VGAE for downstream node-wise learning tasks. Specifically, we train a generative model whose representations are enforced to be globally relevant. This enables increased utilization of the model capacity to model greater data variability and hence mitigates over-pruning.

GAE. Inspired by previous successful unsupervised architecture [30], our GAE uses a two-layer linear projection applied to the learnable linear transformation, W, and high-level representations, H, to produce the feature representation, Z, such as:

$$Z = \mathcal{F}(H, W) \tag{3}$$

The feature representations are then processed using a two-layer linear projection, which takes as an input the latent representations, Z, and learnable projection matrix W:

$$H' = \mathcal{G}(Z, W) \tag{4}$$

We directly measure the Euclidean discrepancy between the high-level node representations, H, and the output of the decoder, H', as the loss

function. The network is trained by reducing the following least-square of the reconstruction loss:

$$\mathcal{L}_r = \|H - H'\|_2^2 \tag{5}$$

Given the latent features, we employ a Student t-distribution with one degree of freedom to predict assignment of clusters using the following equation:

$$p_{ij} = \frac{(1 + \|z_i - \mu_j\|^2)^{-1}}{\sum_{k \neq 1} (1 + \|z_k - \mu_1\|^2)^{-1}} \tag{6}$$

where p_{ij} represents the probability of node i being in cluster j and μ_j is the centroid of cluster j. Our clustering objective is to reduce the difference between the probability distributions of the targets H' and predictions P using the following Kullback Leibler (KL) divergence:

$$\mathcal{L}_c = \mathcal{KL}(H'||P) = \sum_{i=1}^{N} \sum_{j=1}^{K} h'_{ik} \log \frac{h'_{ik}}{p_{ik}} \tag{7}$$

with

$$h'_{ik} = \frac{p_{ik} / \left(\sum_{i'} p_{i'k}\right)^{\frac{1}{2}}}{\sum_{k'} p_{ik'} / \left(\sum_{i'} p_{i'k'}\right)^{\frac{1}{2}}} \tag{8}$$

SOLI-GAE objective is defined as a sum of the reconstruction loss which is used to learn representations that can preserve intrinsic local structure in data and the clustering loss which is responsible for manipulating embedded space in order to scatter embedded points:

$$\mathcal{L} = \mathcal{L}_r + \lambda \mathcal{L}_c \tag{9}$$

where $\mathcal{L}_r$ and $\mathcal{L}_c$ are reconstruction loss and clustering loss respectively, and $\lambda > 0$ controls the degree of distorting embedded space.

VGAE. For the variational autoencoder model, we follow intuitions from VGAE [9] and employ a Gaussian prior parametrized by a two-layer GCN as

our inference model:

$$q\left(Z|H,A'\right)=\prod_{i=1}^{N}q\left(z_i|H,A'\right),\quad \text{with}\quad q\left(z_i|H,A'\right)=\mathcal{N}\left(z_i|\mu_i,diag\left(\sigma_i^2\right)\right) \tag{10}$$

where the mean vector, μ_i, and the variance vector, $\log\sigma_i$, are the i^{th} row of the output of a two layer GCN and $GCN_\mu\left(H,A'\right)$ and $GCN_\sigma\left(H,A'\right)$ share the weight parameters of the first layer, W_0. Our generative model is a simple inner product between hidden representations:

$$p\left(A''|Z\right)=\prod_{i=1}^{N}\prod_{j=1}^{N}p\left(A''_{ij}|z_i,z_j\right),\quad \text{with}\quad p\left(A''_{ij}=1|z_i,z_j\right)=\sigma\left(z_i^Tz_j\right), \tag{11}$$

where σ is the logistic sigmoid nonlinearity. VGAE model is trained using backpropagation to minimize the loss, $\mathcal{L}$, by balancing between explaining the data (first term) and ensuring that the posterior distribution is close to the prior $p(Z)$ (second term) as follow:

$$\mathcal{L}=\mathbb{E}_q(Z|H,A')\left[\log p\left(A''|Z\right)\right]-\lambda\mathcal{KL}\left[q\left(Z|H,A'\right)||p\left(Z\right)\right], \tag{12}$$

where λ manages the significance of keeping the information encoded in Z; $\lambda=0$ leads to a vanilla autoencoder while $\lambda=1$ corresponds to the VGAE objective. For every sparse adjacency matrix, we re-weight terms with $A'_{ij}=1$ in $\mathcal{L}$. We further take a Gaussian prior $p(Z)=\prod_i p(z_i)=\prod_i\mathcal{N}(z_i|0,I)$.

Assuming the single-graph setup (i.e., (X, A) provided as input), we now summarize the steps of the SOLI procedure:

1. Find all maximal cliques from the original graph by passing it through the clique function: $A'=\mathcal{Q}(A)$.
2. Sample negative examples by using the corruption function: $\left(\widetilde{X},\widetilde{A'}\right)\sim\mathcal{C}\left(X,A'\right)$.
3. Obtain high-level representations, $\vec{h_i}$, for the input graph by passing it through the encoder: $H=\mathcal{E}(X,A')=\{\vec{h_1},\vec{h_2},\ldots,\vec{h_N}\}$.
4. Obtain high-level representations, $\widetilde{\vec{h_{ij}}}$, for the negative examples by passing them through the encoder: $\widetilde{H_i}=\mathcal{E}(\widetilde{X},\widetilde{A'})=\{\widetilde{\vec{h_{i1}}},\widetilde{\vec{h_{i2}}},\ldots,\widetilde{\vec{h_{iN}}}\}$.

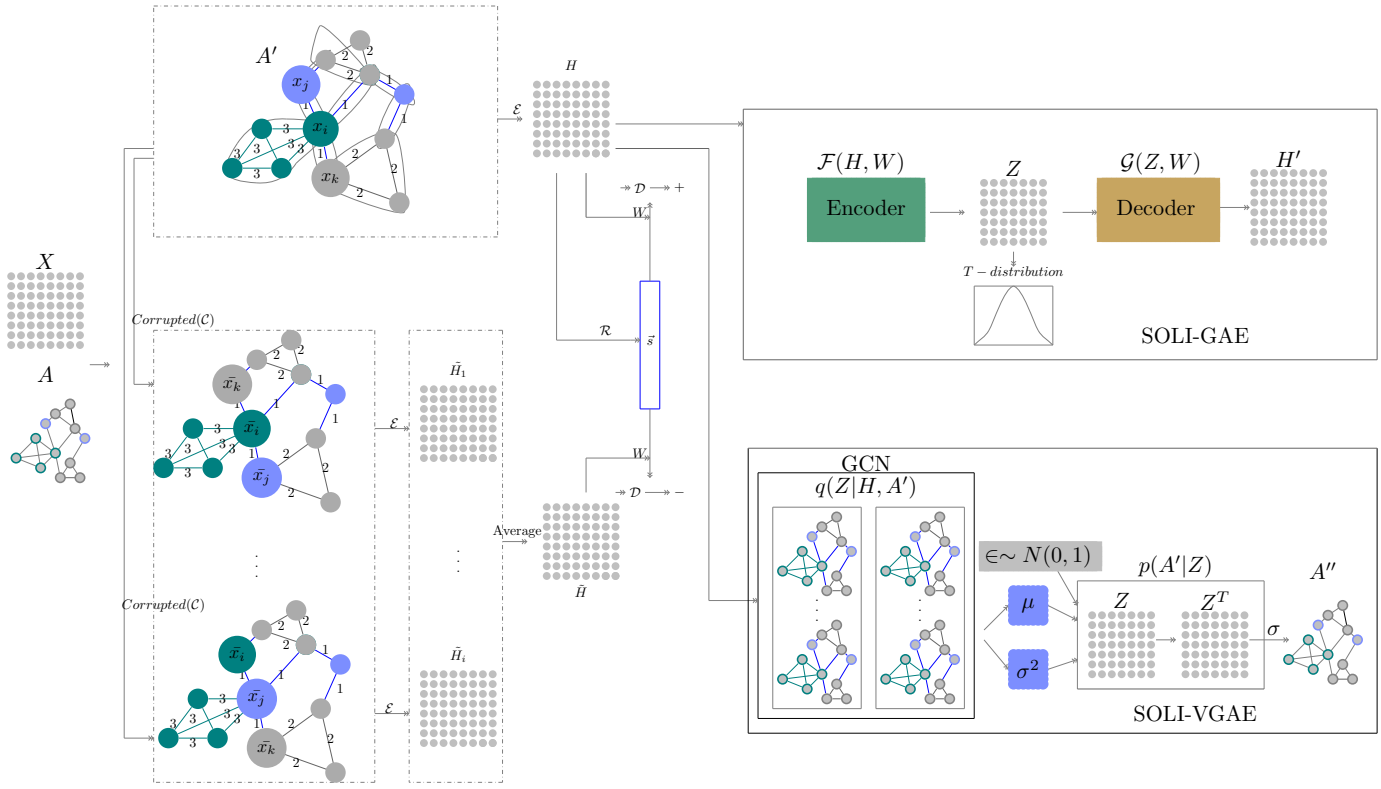


Figure 2: SOLI learns representations and evaluates the node-level clustering utility of these representations in a fully unsupervised manner.

5. Take an average of patch representations of the negative samples by passing them through the average function: $\widetilde{H} = avg(\widetilde{H_i})$.
6. Summarize patch representations of the input graph by passing it through the readout function: $\vec{s} = \mathcal{R}(H)$.
7. Update parameters of $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{D}$ by applying gradient descent to maximize Equation (2).
8. Obtain latent representations, Z, by passing high-level representations through either GAE (or VGAE) encoder: $Z = \mathcal{F}(H)$.
9. Reconstruct high-level representations by passing latent representations through GAE (or VGAE) decoder: $H' = \mathcal{G}(Z)$.
10. Update parameters of Q and P by applying gradient descent to minimize Equation (7) for GAE or optimize λ parameter to balance Equation (12) for VGAE.

This algorithm is graphically summarized in Figure 2.

5. Experiments

Datasets. Our experiments were conducted on Cora, Citeseer, and Pubmed citation network benchmark datasets [31]. A brief statistics of these datasets can be found in Table 1. Nodes correspond to scientific papers and (undirected) edges to citations in all of these datasets, i.e. when a document cites another document, there will be an undirected link between them. Features represent bag-of-words components of a research paper. Cora, Citeseer, and Pubmed datasets are divided into seven, six, and three classes respectively.

Table 1: Statistics of the citation network datasets.

Dataset	Nodes	Edges	Features	Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
PubMed	19,717	44,338	500	3

Baselines. Representations learned by SOLI have been assessed against Spectral Clustering (SC) [32], DeepWalk (DW) [1], GAE [9], VGAE [9], ARGA [18], ARVGA [18], RWR-GAE [19], and RWR-VGAE [19].

Tasks. Following intuitions from [33], we utilize the following metrics to assess the performance of the SOLI on the clustering task: Average rand index (ARI), normalized mutual information (NMI), precision, F-score (F1), and accuracy (Acc). Following intuitions from [9], we evaluate the average precision (AP) and the area under a receiver operating characteristic curve (AUC) to validate the performance of our strong baselines on the link prediction task.

Settings. Our model is initialized using Glorot initialization [34] and trained with an initial learning rate of 0.01 for a maximum of 200 epochs using Adam SGD optimizer [35]. We terminate the training if the validation accuracy does not improve for 10 consecutive steps; as a result, most runs finish in less than 200 steps. It is applied a fixed dropout [36] rate of 0.7 to the input and hidden layers. A $\mathcal{L}_2$ regularization of 0.0005 on the weights is also considered. We use a one-layer GCN model as an encoder with an effective hidden size of 512 (especially 256 on Pubmed due to memory limitation) and the parametric ReLU (PReLU) [37] nonlinearity. For training the variational auto-encoder, we use hyper-parameters provided by [9] and apply full-batch gradient descent while making use of the reparameterization trick [17].

Results. Table 2 reports the mean AP and AUC for the link prediction task over 10 runs for each dataset. Our proposed approach demonstrates state-of-the-art performance being achieved across all three datasets for two evaluation metrics. The best results are highlighted in bold.

Tables 3 to 5 demonstrate how our model performs on the node clustering task across three benchmark datasets. We report the mean of the clustering metrics after 10 runs of each experiment. For the Cora dataset, we find that the SOLI-GAE method improves Acc by 25% and 7% comparing to DeepWalk and ARGA respectively. On the Citeseer dataset, SOLI-VGAE outperforms ARVGA by 10% on Acc, F1, and Precision. For the PubMed dataset, SOLI-VGAE outperforms RWR-VGAE by 6% and 8% on ARI and NMI respectively.

Table 2: Performance comparison of different models for the Link Prediction task

Model	**Cora**		**Citeseer**		**Pubmed**	
	AUC	AP	AUC	AP	AUC	AP
SC	84.6	88.5	80.5	85.0	84.2	87.8
DW	83.1	85.0	80.5	83.6	84.4	84.1
GAE	91.0	92.0	89.5	89.9	96.4	96.5
VGAE	91.4	92.6	90.8	92.0	94.4	94.7
ARGA	92.4	93.2	91.9	93.0	96.8	97.1
ARVGA	92.4	92.6	92.4	93.0	96.5	96.8
RWR-GAE	92.9	92.7	92.1	91.5	96.2	96.3
RWR-VGAE	92.6	92.7	92.3	92.4	95.3	95.2
SOLI-GAE	**99.4**	**99.3**	**98.3**	**98.9**	**98.0**	**97.2**
SOLI-VGAE	94.4	94	95.4	94.3	96.9	96.8

Table 3: Performance comparison of different models for the Clustering task on Cora.

Model	**Acc**	**NMI**	**F1**	**Precision**	**ARI**
SC	0.367	0.127	0.318	0.193	0.031
DW	0.484	0.327	0.392	0.361	0.243
GAE	0.596	0.374	0.595	0.596	0.274
VGAE	0.625	0.371	0.625	0.625	0.319
ARGA	0.668	0.489	0.668	0.668	0.422
ARVGA	0.544	0.433	0.544	0.544	0.310
RWR-GAE	0.593	0.431	0.577	0.577	0.341
RWR-VGAE	0.577	0.431	0.577	0.577	0.372
SOLI-GAE	**0.739**	**0.56**	**0.737**	**0.730**	**0.506**
SOLI-VGAE	0.726	0.546	0.726	0.726	0.520

Table 4: Performance comparison of different models for the Clustering task on Citeseer.

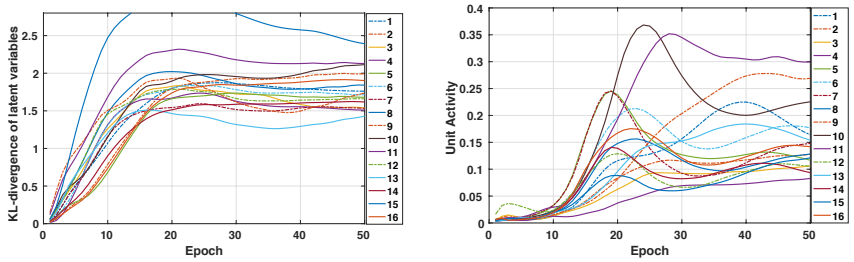
Model	Acc	NMI	F1	Precision	ARI
SC	0.239	0.056	0.299	0.179	0.031
DW	0.337	0.088	0.270	0.248	0.243
GAE	0.412	0.142	0.412	0.412	0.097
VGAE	0.391	0.133	0.391	0.391	0.070
ARGA	0.508	0.269	0.508	0.508	0.214
ARVGA	0.597	0.339	0.597	0.597	0.330
RWR-GAE	0.440	0.252	0.440	0.440	0.180
RWR-VGAE	0.519	0.289	0.519	0.519	0.257
SOLI-GAE	0.682	0.408	0.682	0.682	0.400
SOLI-VGAE	**0.698**	**0.442**	**0.698**	**0.698**	**0.457**

Table 5: Performance comparison of different models for the Clustering task on Pubmed.

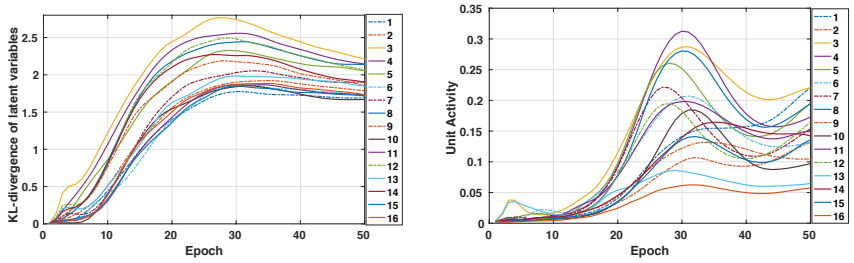
Model	Acc	NMI	F1	Precision	ARI
GAE	0.672	0.224	0.672	0.672	0.245
VGAE	0.673	0.225	0.673	0.673	0.245
ARGA	0.618	0.214	0.618	0.618	0.197
ARVGA	0.418	0.045	0.418	0.418	0.019
RWR-GAE	0.664	0.268	0.651	0.681	0.267
RWR-VGAE	0.672	0.264	0.672	0.673	0.273
SOLI-GAE	0.702	0.325	0.70	0.707	0.320
SOLI-VGAE	**0.706**	**0.345**	**0.702**	**0.732**	**0.337**

Figure 3 illustrates the activity level and $\mathcal{KL}$-divergence in VGAE, RWR-VGAE, and SOLI-VGAE for each component of a 16-unit VAE trained on Cora dataset. Following [38], we define a unit as an active, if $A_u = \mathrm{Cov}_x(\mathbb{E}_{u\sim q(u|x)}[u]) > 0.02$. While all the hidden units in VGAE, RWR-VGAE, and SOLI-VGAE are active, the value of $\mathcal{KL}$-divergence for all latent variables in VGAE and RWR-VGAE is quite high (1.5 and higher), revealing negligible matching of the posterior distribution with the prior. This negatively influences the models' generative capacity. We observe that in SOLI-VGAE, latent variables have $\mathcal{KL}$-divergence of 1.5 and lower, indicating that

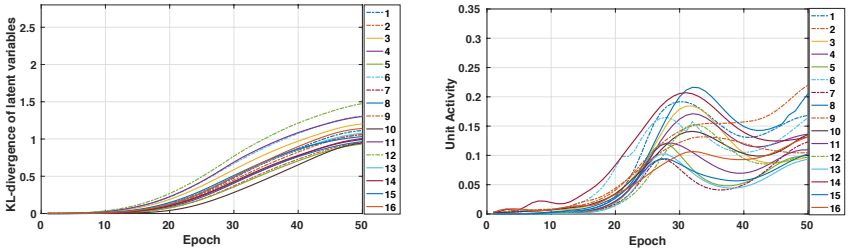
they are highly correlated with standard Gaussian distribution.



(a) $\mathcal{KL}$-divergence of latent variables in VGAE (b) Unit activity of 16 hidden units in VGAE



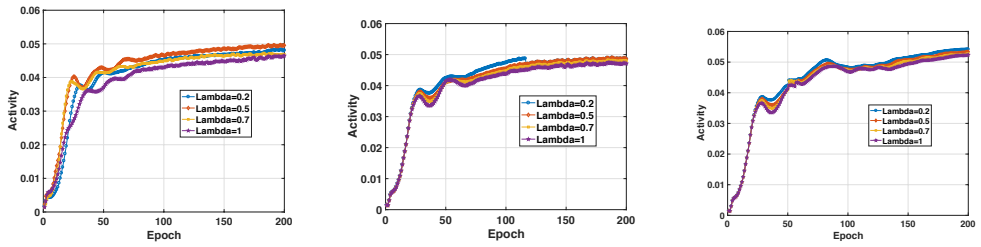
(c) $\mathcal{KL}$-divergence of latent variables in RWR-VGAE (d) Unit activity of 16 hidden units in RWR-VGAE



(e) $\mathcal{KL}$-divergence of latent variables in SOLI-VGAE (f) Unit activity of 16 hidden units in SOLI-VGAE

Figure 3: Unit activity and $\mathcal{KL}$ term for a 16-unit VGAE, RWR-VGAE, and SOLI-VGAE. SOLI-VGAE is able to utilize the full latent capacity with all units active.

Figure 4 shows the effect of λ on unit activity in VGAE, RWR-VGAE, and SOLI-VGAE on Cora dataset. When $\lambda = 1$, the activity is at its lowest level. Unlike VGAE that has its highest activity at $\lambda = 0.5$, RWR-VGAE and SOLI-VGAE reach their maximum activity at $\lambda = 0.2$.

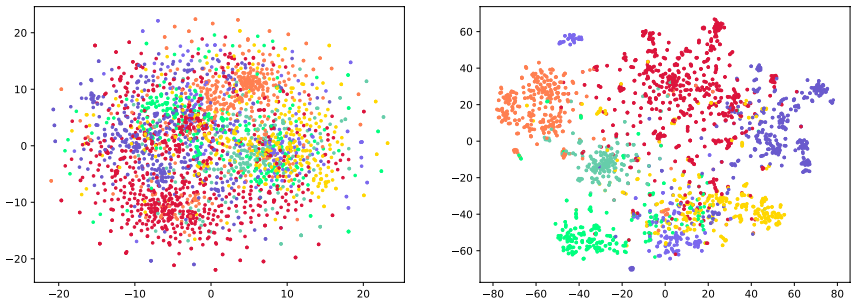


(a) VGAE (b) RWR-VGAE (c) Soli-VGAE

Figure 4: Activity of latent units in VGAE, RWR-VGAE, and SOLI-VGAE on Cora, shown for varying values of the $\mathcal{KL}$ weight λ.

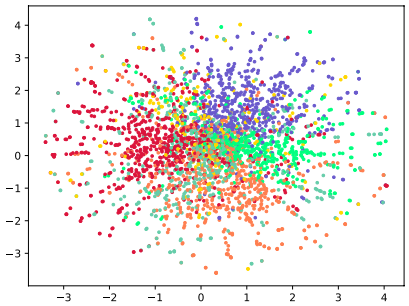
6. Qualitative Analysis

To better understand the effectiveness of SOLI, we provide a range of standard t-SNE plots [39] of the representations learned by the SOLI-GAE algorithm on the Cora, CiteSeer, and Pubmed dataset in Figure 5. Colors denote document class. Compared to the raw features, the learned representations, projected in the 2D space, exhibit discernible clustering, which respects the number of topic classes in each dataset.

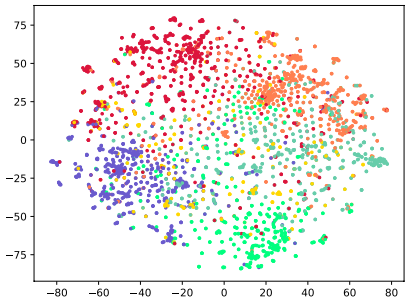


(a) Cora Raw Features

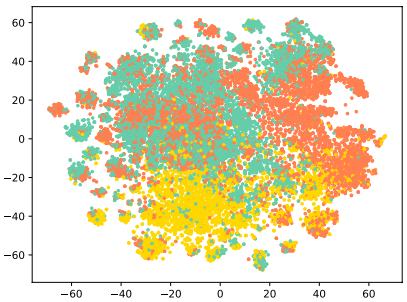
(b) Cora Learned Features



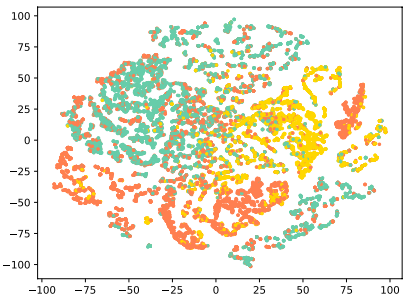
(c) CiteSeer Raw Features



(d) CiteSeer Learned Features



(e) Pubmed Raw Features



(f) Pubmed Learned Features

Figure 5: t-SNE embeddings of the nodes in the Cora, CiteSeer, and Pubmed datasets from the raw features (**left**) and features from a learned model (**right**). The clusters of the learned SOLI-GAE model's representations are clearly defined.

7. Conclusions

This paper introduces a model-based approach for unsupervised graph representation learning. By maximizing local mutual information across

maximal cliques, we are able to learn patch representations that are mindful of the globally relevant graph information. By facilitating more latent variables to actively play their part in the reconstruction, our proposed approach mitigates the issue of over-pruning. By assessing the effectiveness of the learned patch representations on a range of node-wise learning tasks across three well-established benchmark datasets, we verified the performance of our approach.

References

[1] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2014, pp. 701–710.

[2] Q. Huang, Y. Zhang, H. Peng, T. Dan, W. Weng, H. Cai, Deep subspace clustering to achieve jointly latent feature extraction and discriminative learning, Neurocomputing 404 (2020) 340–350.

[3] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 974–983.

[4] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, J. Leskovec, Hierarchical graph representation learning with differentiable pooling, in: Advances in neural information processing systems, 2018, pp. 4800–4810.

[5] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).

[6] S. Molaei, H. Zare, H. Veisi, Deep learning approach on information diffusion in heterogeneous networks, Knowledge-Based Systems 189 (2020) 105153.

[7] Q. Wang, Z. Meng, X. Li, Locality adaptive discriminant analysis for spectral–spatial classification of hyperspectral images, IEEE Geoscience and Remote Sensing Letters 14 (11) (2017) 2077–2081.

[8] T. R. Davidson, L. Falorsi, N. De Cao, T. Kipf, J. M. Tomczak, Hyperspherical variational auto-encoders, arXiv preprint arXiv:1804.00891 (2018).

[9] T. N. Kipf, M. Welling, Variational graph auto-encoders, arXiv preprint arXiv:1611.07308 (2016).

[10] X. Li, M. Chen, F. Nie, Q. Wang, A multiview-based parameter free framework for group detection, in: Thirty-First AAAI Conference on Artificial Intelligence, 2017.

[11] P. Cui, X. Wang, J. Pei, W. Zhu, A survey on network embedding, IEEE Transactions on Knowledge and Data Engineering 31 (5) (2018) 833–852.

[12] A. Bojchevski, S. Günnemann, Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking, arXiv preprint arXiv:1707.03815 (2017).

[13] S. Yeung, A. Kannan, Y. Dauphin, L. Fei-Fei, Tackling over-pruning in variational autoencoders, arXiv preprint arXiv:1706.03643 (2017).

[14] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, S. Bengio, Generating sentences from a continuous space, arXiv preprint arXiv:1511.06349 (2015).

[15] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2016, pp. 1225–1234.

[16] D. Charte, F. Charte, M. J. del Jesus, F. Herrera, An analysis on the use of autoencoders for representation learning: Fundamentals, learning task case studies, explainability and challenges, Neurocomputing (2020).

[17] D. P. Kingma, M. Welling, Auto-encoding variational bayes, arXiv preprint arXiv:1312.6114 (2013).

[18] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, C. Zhang, Adversarially regularized graph autoencoder for graph embedding, arXiv preprint arXiv:1802.04407 (2018).

[19] P.-Y. Huang, R. Frederking, et al., Rwr-gae: Random walk regularization for graph auto encoders, arXiv preprint arXiv:1908.04003 (2019).

[20] P. Goyal, E. Ferrara, Graph embedding techniques, applications, and performance: A survey, Knowledge-Based Systems 151 (2018) 78–94.

[21] A. Tsitsulin, D. Mottin, P. Karras, E. Müller, Verse: Versatile graph embeddings from similarity measures, in: Proceedings of the 2018 World Wide Web Conference, 2018, pp. 539–548.

[22] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781 (2013).

[23] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2016, pp. 855–864.

[24] S. Cao, W. Lu, Q. Xu, Grarep: Learning graph representations with global structural information, in: Proceedings of the 24th ACM international on conference on information and knowledge management, 2015, pp. 891–900.

[25] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, J. Tang, Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 2018, pp. 459–467.

[26] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, R. D. Hjelm, Deep graph infomax, arXiv preprint arXiv:1809.10341 (2018).

[27] A. Douik, H. Dahrouj, T. Y. Al-Naffouri, M.-S. Alouini, A tutorial on clique problems in communications and signal processing, arXiv preprint arXiv:1808.07102 (2018).

[28] C. Bron, J. Kerbosch, Algorithm 457: finding all cliques of an undirected graph, Communications of the ACM 16 (9) (1973) 575–577.

[29] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, Y. Bengio, Learning deep representations by mutual information estimation and maximization, arXiv preprint arXiv:1808.06670 (2018).

[30] X. Guo, L. Gao, X. Liu, J. Yin, Improved deep embedded clustering with local structure preservation.

[31] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad, Collective classification in network data, AI magazine 29 (3) (2008) 93.

[32] L. Tang, H. Liu, Leveraging social media networks for classification, Data Mining and Knowledge Discovery 23 (3) (2011) 447–478.

[33] R. Xia, Y. Pan, L. Du, J. Yin, Robust multi-view spectral clustering via low-rank and sparse decomposition, in: Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014.

[34] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the thirteenth international conference on artificial intelligence and statistics, 2010, pp. 249–256.

[35] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research 15 (1) (2014) 1929–1958.

[37] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.

[38] Y. Burda, R. Grosse, R. Salakhutdinov, Importance weighted autoencoders, arXiv preprint arXiv:1509.00519 (2015).

[39] L. v. d. Maaten, G. Hinton, Visualizing data using t-sne, Journal of machine learning research 9 (Nov) (2008) 2579–2605.