Graph-Wise Common Latent Factor Extraction for Unsupervised Graph Representation Learning

Thilini Cooray, Ngai-Man Cheung

Singapore University of Technology and Design (SUTD)
thilini_cooray@mymail.sutd.edu.sg, ngaiman_cheung@sutd.edu.sg

Abstract

Unsupervised graph-level representation learning plays a crucial role in a variety of tasks such as molecular property prediction and community analysis, especially when data annotation is expensive. Currently, most of the best-performing graph embedding methods are based on Infomax principle. The performance of these methods highly depends on the selection of negative samples and hurt the performance, if the samples were not carefully selected. Inter-graph similarity-based methods also suffer if the selected set of graphs for similarity matching is low in quality. To address this, we focus only on utilizing the current input graph for embedding learning. We are motivated by an observation from real-world graph generation processes where the graphs are formed based on one or more global factors which are common to all elements of the graph (e.g., topic of a discussion thread, solubility level of a molecule). We hypothesize extracting these common factors could be highly beneficial. Hence, this work proposes a new principle for unsupervised graph representation learning: Graph-wise Common latent Factor EXtraction (GCFX). We further propose a deep model for GCFX, deepGCFX, based on the idea of reversing the above-mentioned graph generation process which could explicitly extract common latent factors from an input graph and achieve improved results on downstream tasks to the current state-of-the-art. Through extensive experiments and analysis, we demonstrate that, while extracting common latent factors is beneficial for graph-level tasks to alleviate distractions caused by local variations of individual nodes or local neighbourhoods, it also benefits node-level tasks by enabling long-range node dependencies, especially for disassortative graphs.

Introduction

Graph structured data has been very useful in representing a variety of data types including social networks (Newman and Girvan 2004), protein-protein interactions (Krogan et al. 2006), scene graphs (Krishna et al. 2017), customer purchasing patterns (Yada et al. 2004) and many more (Pang and Cheung 2017; Cooray, Cheung, and Lu 2020; Liu and Cheung 2021). In this work, we focus on graph-level representation learning. It is crucial for tasks like molecular property identification (Duvenaud et al. 2015) and community classification (Yanardag and Vishwanathan 2015), and they are useful for applications such as drug discovery, material design and recommendation systems. Availability of task-specific labels plays a significant role in graph representation learning. However, annotations are very expensive (Yang et al. 2019; Wieder et al. 2020; Sun et al. 2020) due to many specialized fields in which graphs are utilized (e.g., biological sciences, quantum mechanics). Therefore, unsupervised graph representation learning has become crucial.

Unsupervised graph level representation learning has a very rich literature consisting of several main directions. Skip-gram influenced graph embedding methods (node2vec (Grover and Leskovec 2016), sub2vec (Adhikari et al. 2018), graph2vec (Narayanan et al. 2017)) only rely on neighbourhood information and lose the advantage of using node features, making them less effective. Kernel methods (Random Walk (RW) (Gärtner, Flach, and Wrobel 2003), Shortest Path (SP) (Borgwardt and Kriegel 2005), Graphlet Kernel (GK) (Shervashidze et al. 2009), DDGK (Al-Rfou, Perozzi, and Zelle 2019), GCKN (Chen, Jacob, and Mairal 2020)) and graph proximity methods (UGraphEmb (Bai et al. 2019)) use pair-wise inter-graph similarity calculations making them more effective, but less efficient. Quality of learnt embeddings by this method heavily relies on the quality and variety of other graphs with which it compares. Contrastive learning (InfoGraph (Sun et al. 2020), CMV (Multi-view) (Hassani and Khasahmadi 2020), GCC (Extra data) (Qiu et al. 2020) and GraphCL(Augmentations) (You et al. 2020)) is the newest addition which is based on the Infomax principle (Linsker 1988) which aims at obtaining an output which has maximum mutual information with the input. The main drawback of these methods (Grill et al. 2020) is that their heavy reliance on the selection procedure of negative samples for model performance. A careful selection of task-wise negative samples is required to obtain good performance. While both inter-graph similarity and contrastive methods achieve state-of-the-art for graph embedding learning, both of them suffer very high if the quality of other graphs they compare with is low.

Autoencoder (Baldi and Hornik 1989; Hinton and Zemel 1993) based embedding methods solve this weakness by only utilizing its current input for representation learning. However, existing graph autoencoder models (Kipf and Welling 2016; Pan et al. 2018; Park et al. 2019) are only aimed at node-level modelling. **In particular, these meth-**

ods are unable to extract graph-wise common latent factors. Other issues of these methods are their overemphasis on proximity information (Hassani and Khasahmadi 2020) and the inability to differentiate features (Tian, Krishnan, and Isola 2020). Feature differentiation is very important when using embeddings in downstream tasks because equally treated features could add noise and redundancy which leads to performance degradation.

These weaknesses of existing work motivate us to research an approach that could both differentiate features crucial for graph-level representations as well as capable of learning embeddings by only utilizing the current input sample. Although GVAE (Kipf and Welling 2016) is inadequate to fulfil graph-level feature differentiation, it empowers single sample-based learning. Hence, we are motivated to follow a generative-based mechanism while addressing the specific requirements to obtain a discriminative graph representation which GVAE lacks on.

Graph-wise common latent factor extraction (GCFX) Motivation. To draw inspiration for our generative-based approach, we observe two real-world graph formation examples. An online discussion thread can be represented as a graph where nodes represent users who have participated in the discussion thread, and edges represent interactions among users in the thread (Yanardag and Vishwanathan 2015). This graph initializes with a single user who wants to discuss a particular topic and grows with nodes when subsequent users start responding about this topic. For the second example, a chemical compound can be represented as a graph where the nodes are atoms and edges are chemical bonds. Inverse molecule design (Sanchez-Lengeling and Aspuru-Guzik 2018; Kuhn and Beratan 1996; Zunger 2018) is a molecule generation process that is initiated with the desired properties to be included in a molecule such as solubility and toxicity levels. De novo (Schneider 2013; Brown et al. 2019) inverse molecule design method initiates with the desired ranges of those properties and iteratively adds atoms and chemical bonds conditioned on those properties to form molecular graphs.

The key observation from these examples is that each node and edge added to the graph was conditioned on one or more common graph factors. The topic is the common global factor for all elements of the discussion thread and toxicity and solubility levels are common for the entire chemical compound. We can see that although each node has its own specific information such as personal details of a user or properties of an atom, common factors cause the differentiation of one graph from another. Hence, useful for tasks like community detection and molecule selection for drug discovery.

Graph-wise common latent factor extraction. Motivated by this, we hypothesize extracting these common factors could be highly beneficial for a discriminative graph representation. Hence, this work proposes graph-wise common factor extraction in a latent manner. We further propose deepGCFX: a novel autoencoder-based architecture that can explicitly extract common latent factors from the entire graph incorporating feature differentiation to autoencoders. Our enhanced decoding mechanism which enforces utilizing common latent factors for graph reconstruction, also regularizes normal autoencoder's heavy dependence on proximity.

We summarize our contributions as follows:

- We propose GCFX: a novel principle for unsupervised graph representation learning based on the notion of graph-wise common latent factors inspired by real-world examples.
- Existing autoencoder models are unable to learn graph-wise common factors due to their inability of feature differentiation. Therefore, we propose deepGCFX[1] : a novel autoencoder-based approach with iterative query-based reasoning and feature masking capability to extract common latent factors.
- We empirically demonstrate the effectiveness of deepGCFX in extracting graph-wise common latent factors.
- To the best of our knowledge, this is the first graph embedding learning method based on GCFX. We show that deepGCFX can achieve state-of-the-art results in *unsupervised graph-level representation learning* as shown in standard downstream tasks.
- By extracting common factors from non-common, deepGCFX enables long-distance inter-node information sharing ability for node representation learning achieving best results for unsupervised node representation learning on disassortative graphs.

Methodology

Graph Generation Process

Let $\mathbb{D} = \{\mathbb{G}, C_f, L_f\}$ be the set that consists of graphs and their ground truth common and uncommon generative factors. We would call uncommon factors as local, hence the notation L. Each graph $G = (V, A)$, contains a set of nodes V and A is the adjacency matrix. C_f and L_f represent two sets of generative factors: C_f contains common factors $\mathbf{c_f} \subset C_f$ common for the entire graph (e.g., discussion topic) and $\mathbf{l_f} \subset L_f$ represents local factors which can differ from local patch to patch (e.g., user information). In this work we assume, $\mathbf{c_f}$ and $\mathbf{l_f}$ are conditionally independent given G, where $p(\mathbf{c_f}, \mathbf{l_f}|G) = p(\mathbf{c_f}|G) \cdot p(\mathbf{l_f}|G)$. We assume that the graph G is generated using a true world generator which uses the ground truth generative factors: $p(G|\mathbf{c_f}, \mathbf{l_f}) = \mathit{Gen}(\mathbf{c_f}, \mathbf{l_f})$.

GCFX: Graph-Wise Common Latent Factor Extraction

We focus on the novel problem of graph-wise common factor extraction. Although we only focus on extracting common latent factors, identifying local factors is essential to filter them out. Hence, our goal is to develop an unsupervised deep graph generative model which can learn the joint distribution of graph G, the set of generative factors $\mathbf{Z}$, using only the samples from $\mathbb{G}$. This should be learnt in a way that the set of latent generative factors can generate the observed graph G, such that $p(G|\mathbf{Z}) \approx p(G|\mathbf{c_f}, \mathbf{l_f}) = \mathit{Gen}(\mathbf{c_f}, \mathbf{l_f})$. A

[1]Our source code: https://github.com/thilinicooray/deepGCFX

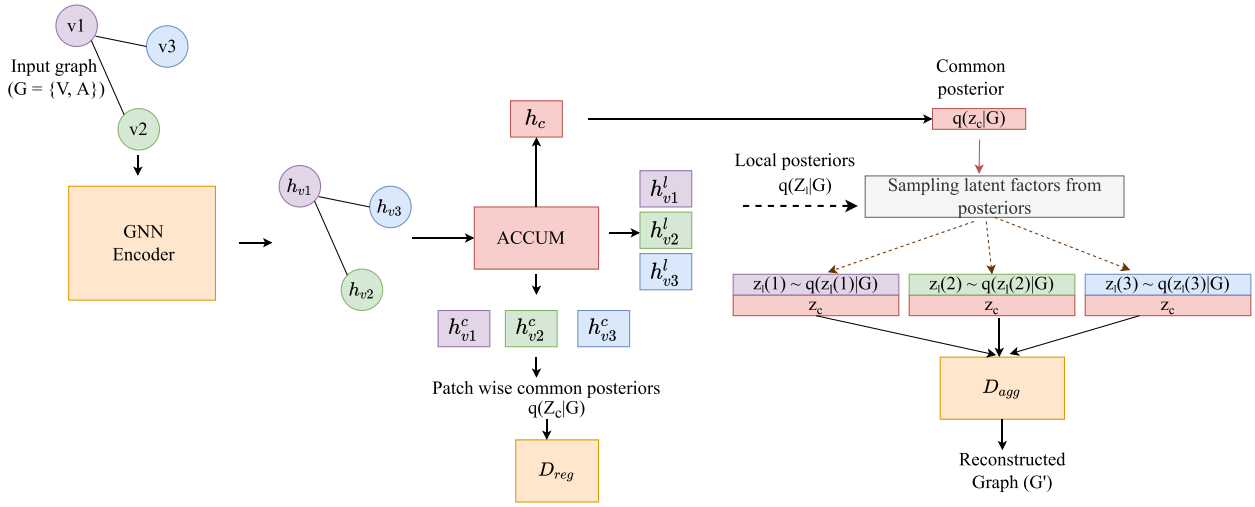


Figure 1: deepGCFX architecture: An input graph G is first sent through a GNN Encoder to obtain individual node representations aggregated with neighbours and then the ACCUM procedure is conducted to filter graph-wise common ($\mathbf{h}^c$) and local factors ($\mathbf{h}^l$) from each patch to obtain a single graph-wise common latent factor representation $\mathbf{h}_c$ (ACCUM is described Fig. 2). Sampled local latent factors from their respective posteriors $\mathbf{z}_l(j) \sim q(\mathbf{z}_l(j)|G),\ \forall j \in \{1 \dots |V|\}$ are combined with the common latent $\mathbf{z}_c \sim q_(\mathbf{z}_c|G)$ and input to the decoder $\mathcal{D}_{agg}$ to reconstruct G. $\mathcal{D}_{reg}$ is used to enforce $\mathbf{z}_c$ to contain G related factors. deepGCFX is optimized using the loss function in Eq.17.

suitable approach to fulfil this objective is to maximize the marginal log-likelihood for the observed graph G over the whole distribution of latent factors $\mathbf{Z}$.

$$\max_{\theta} \mathbb{E}_{p_\theta(\mathbf{Z})}[\log p_\theta(G|\mathbf{Z})] \tag{1}$$

For an observed graph G, the inferred posterior probability distribution of the latent factors $\mathbf{Z}$ can be described as $q_\phi(\mathbf{Z}|G)$. However, the graph generation process we described above assumes two independent sets of generative factors representing common and local information of a graph, from which we are explicitly interested in common factors. Therefore, we consider a model where the latent factor set $\mathbf{Z}$ can be divided into two independent latent factor sets as $\mathbf{Z} = (\mathbf{Z}_c, \mathbf{Z}_l)$. $\mathbf{Z}_c$ represents the latent factors that capture the graph-wise common generative factors of G and $\mathbf{Z}_l$ captures its local counterpart. Therefore, we can rewrite our inferred posterior distribution as follows:

$$q_\phi(\mathbf{Z}|G) = q_\phi(\mathbf{Z}_c, \mathbf{Z}_l|G) = q_\phi(\mathbf{Z}_c|G)q_\phi(\mathbf{Z}_l|G) \tag{2}$$

We discuss these two posteriors in detail: $q_\phi(\mathbf{Z}_c|G)$ and $q_\phi(\mathbf{Z}_l|G)$. The graph G consists of $|V|$ number of nodes. In a graph data structure, each node is not isolated. They are connected with their neighbours and propagate information. Therefore, we use the term *patch* to indicate the local neighbourhood centered at each node which the node interacts with. Therefore, $q_\phi(\mathbf{Z}_c|G)$ and $q_\phi(\mathbf{Z}_l|G)$ are the posterior distributions of all these $|V|$ patches. However, common latent posterior is common for all $|V|$ patches, as the graph G was originally generated with $\mathbf{c_f}$ common for all V. Hence, we propose to use a single latent $\mathbf{z}_c$ to capture the common generative factors. In particular, we use $q_\phi(\mathbf{z}_c|G)$ to model this single posterior. On the other hand, the factors which contribute to generate each patch can vary significantly. Therefore, in this model, we assume the local latent factors are independent. Therefore, we update Eq. 2 as:

$$q_\phi(\mathbf{Z}|G) = q_\phi(\mathbf{z}_c, \mathbf{Z}_l|G) = q_\phi(\mathbf{z}_c|G) \prod_{i=1}^{|V|} q_\phi(\mathbf{z}_l(i)|G) \tag{3}$$

Here, $\mathbf{z}_l(i)$ are the latent factors that capture the local generative factors for a patch centered at node i. Now, our objective is to make sure the latent factors sampled from common and local latent posterior distributions can capture the common and local generative factors $\mathbf{c}_f$ and $\mathbf{l}_f$ separately.

A Constrained Optimization Formulation for GCFX

Now we try to match common and local generative factors $\mathbf{c}_f$ and $\mathbf{l}_f$ to their respective priors $p(\mathbf{z}_c)$ and $p(\mathbf{z}_l)$ separately. We select unit Gaussians ($\mathcal{N}(0, 1)$) as priors. Based on our modeling of common and local factors in Eq. 3, we can rewrite Eq.1 as a constrained optimization as follows (Higgins et al. 2017):

$$\begin{aligned} \max_{\theta,\phi} \quad & \mathbb{E}_{G\sim\mathbb{G}}\Big[\mathbb{E}_{q_\phi(\mathbf{z}_c,\mathbf{Z}_l|G)}[\log p_\theta(G|\mathbf{z}_c, \mathbf{Z}_l)]\Big] \\ \text{s.t.} \quad & KL(q_\phi(\mathbf{z}_c|G) \parallel p(\mathbf{z}_c)) < \epsilon \\ & KL(q_\phi(\mathbf{Z}_l|G) \parallel p(\mathbf{Z}_l)) < \eta \end{aligned} \tag{4}$$

where ϵ and η are strengths of each constraint. Following Higgins et al. (2017), Eq.4 can be written to obtain the varia-

tional evidence lower bound (ELBO) of a Graph Variational Autoencoder (GVAE) (Kipf and Welling 2016) (Here we use GVAE because of our input is a graph) as follows with β and γ coefficients:

$$\begin{aligned}\mathcal{F}(\theta,\phi;G,\mathbf{z}_c,\mathbf{Z}_l,\beta,\gamma) &\geq \mathcal{L}(\theta,\phi;G,\mathbf{z}_c,\mathbf{Z}_l,\beta,\gamma)\\ &= \mathbb{E}_{q_\phi(\mathbf{z}_c,\mathbf{Z}_l|G)}[\log p_\theta(G|\mathbf{z}_c,\mathbf{Z}_l)]\\ &- \beta\, KL(q_\phi(\mathbf{z}_c|G)\parallel p(\mathbf{z}_c))\\ &- \gamma\, KL(q_\phi(\mathbf{Z}_l|G)\parallel p(\mathbf{Z}_l))\end{aligned} \tag{5}$$

Based on Eq.3, we can expand the KL divergence term $KL(q_\phi(\mathbf{Z}_l|G)\parallel p(\mathbf{Z}_l))$ and rewrite our objective function for a single graph G as:

$$\begin{aligned}\mathcal{L}(\theta,\phi;G,\mathbf{z}_c,\mathbf{Z}_l,\beta,\gamma) &= \mathbb{E}_{q_\phi(\mathbf{z}_c,\mathbf{Z}_l|G)}[\log p_\theta(G|\mathbf{z}_c,\mathbf{Z}_l)]\\ &- \beta\, KL(q_\phi(\mathbf{z}_c|G)|p(\mathbf{z}_c))\\ &- \gamma \sum_{i=1}^{|V|} KL(q_\phi(\mathbf{z}_l(i)|G)\parallel p(\mathbf{z}_l(i)))\end{aligned} \tag{6}$$

Overall, **the learning objective of GCFX is to maximize this lower bound** for all the graphs in a minibatch $\mathbb{G}_b$ from the full dataset $\mathbb{G}$:

$$\mathcal{L}_{\theta,\phi}(\mathbb{G}_b) = \frac{1}{|\mathbb{G}_b|}\sum_{r=1}^{|\mathbb{G}_b|}\mathcal{L}(\theta,\phi;G_r,\mathbf{z}_c,\mathbf{Z}_l,\beta,\gamma) \tag{7}$$

DeepGCFX: An Autoencoder Based Approach for GCFX

Existing autoencoder models including GVAE cannot learn graph-wise common factors due to their inability to differentiate factors based on importance. Therefore, we propose deepGCFX, a novel GVAE architecture based on the GCFX principle which can extract graph-wise common factors. We propose an iterative query-based mechanism with feature masking to achieve this ability. Fig. 1 depicts the proposed deep Graph-wise Common Factor Extractor (deepGCFX) model. We utilize N-layer Graph neural Network(GNN)(Kipf and Welling 2017) as the encoder. n^{th} layer of a GNN can be defined in general as

$$\mathbf{a}_v^{(n)} = \text{AGGREGATE}^{(n)}\left(\left\{\mathbf{h}_u^{(n-1)} : u \in \mathcal{N}(v)\right\}\right) \tag{8}$$

$$\mathbf{h}_v^{(n)} = \text{COMBINE}^{(n)}\left(\mathbf{h}_v^{(n-1)}, \mathbf{a}_v^{(n)}\right) \tag{9}$$

where $\mathbf{h}_v^{(n)}$ is the feature vector of a patch centered at node $v \in V$ at the n^{th} layer after propagating information from its neighbours $u \in \mathcal{N}(v)$, where $(v,u) \in A$. $\mathbf{h}_v^{(0)}$ is often initialized with node features. We use the term GNN to indicate any network which uses layers described in Eq. 9. The neighbourhood aggregation function AGGREGATE and node update function COMBINE differs for each specific GNN architecture (Kipf and Welling 2017; Velickovic et al. 2018; Xu et al. 2019).

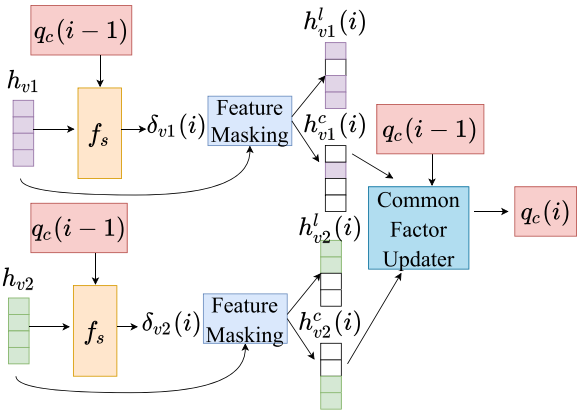


Figure 2: ACCUM : Our main algorithmic invention to enable extraction of high-quality common latent factors for deepGCFX is based on query-based reasoning with feature masking. For this two node graph, at iteration i, the single common factor embedding from previous iteration used as the query $\mathbf{q}_c(i-1)$. For each node, $\delta(i)$ is calculated (Eq. 10) to determine its factor-wise similarity to current common latent factors. Each node's latent factors are divided into common ($\mathbf{h}^c$) and local ($\mathbf{h}^l$) next, using a $\delta(i)$ based mask (Eq. 11-14). Node-wise $\mathbf{h}^c$ are accumulated using Eq.16 to update graph-wise common latent factors for current iteration.

ACCUM : Iterative Query-Based Reasoning With Feature Masking for Common Latent Accumulation

Our main algorithmic invention is a novel mechanism to extract high-quality common latent factors based on ideas of iterative query-based reasoning and feature masking. As discussed, GVAE cannot extract common latent factors as it cannot differentiate feature importance. To ensure the latent factors sampled from common and local posterior distributions can capture common and local factors separately, we first need a mechanism to differentiate and filter out what are common features from the output node/-patch representations of the GNN encoder. To achieve this, we propose a novel mechanism that iteratively learns graph-wise common factors and extracts them from each patch and accumulates them to generate a single common factor embedding for each graph.

We model the common factor extraction as an iterative query-based reasoning problem, where our query is the accumulated common factor representation of the graph. We use that query on our input patch representations $\mathbf{h}_v^{(n)}$ (superscript will be omitted from here onwards.) to determine which factors from $\mathbf{h}_v$ are similar to the existing common factors and filter them out from non-common factors to update the accumulated common factor representation. At each iteration i, the process starts with the query $\mathbf{q_c}(i-1)$ containing graph-wise common factors extracted from all the patches at iteration $i-1$. It is used to query all patch representations $\mathbf{h}_v, v \in V$ at current iteration i to identify the amount of similarity each factor of $\mathbf{h}_v, v \in V$ has with cur-

rent graph-wise common factors $\mathbf{q_c}(i-1)$ which is also our query. The factor-wise similarity score for current iteration i, $\delta_v(i) \in \mathbb{R}^{d\_hidden}$ is calculated as:

$$\delta_v(i) = \sigma(f_s([\mathbf{h}_v\mathbf{W}_k, \mathbf{q_c}(\mathbf{i-1})\mathbf{W}_q])), \tag{10}$$

where $\mathbf{W}_k, \mathbf{W}_q, \in \mathbb{R}^{d\_hidden \times d\_hidden}$ are projection parameters for query and the keys, f_s is a non-linear network and $[\cdot]$ denotes concatenation. Then we create two masks; mask $\mathbf{m}_v^c(i)$ is to filter out factors of patch v, which are similar to current common factors $\mathbf{q_c}(i-1)$ and mask $\mathbf{m}_v^l(i)$ is to filter out remaining local factors.

$$\mathbf{m}_v^c(i) = 1[\![\sigma(\mathbf{h}_v\mathbf{W}_k) \geq \delta_v(i)]\!], \tag{11}$$

$$\mathbf{m}_v^l(i) = 1[\![\sigma(\mathbf{h}_v\mathbf{W}_k) < \delta_v(i)]\!], \tag{12}$$

$$\mathbf{h}_v^c(i) = \mathbf{m}_v^c(i) \odot \mathbf{h}_v\mathbf{W}_v, \tag{13}$$

$$\mathbf{h}_v^l(i) = \mathbf{m}_v^l(i) \odot \mathbf{h}_v\mathbf{W}_v \tag{14}$$

where $\mathbf{W}_v \in \mathbb{R}^{d\_hidden \times d\_hidden}$ is projection parameters and $\mathbf{m}_v^c(i), \mathbf{m}_v^l(i) \in \mathbb{R}^{d\_hidden}$. $\odot$ denotes element-wise multiplication. Now $\mathbf{h}_v^c(i), \forall v \in V$ are accumulated and used to update $\mathbf{q_c}(i-1)$ with newly identified common factors for iteration i using a Gated Recurrent Unit(GRU):

$$\mathbf{q}_{update}(i) = \sum_{v \in V} \mathbf{h}_v^c(i), \tag{15}$$

$$\mathbf{q_c}(i) = \text{GRU}(\mathbf{q}_{update}(i), \mathbf{q_c}(i-1)) \tag{16}$$

This accumulation approach is depicted in Fig. 2. Once ACCUM is over in M iterations, we use $\mathbf{h}_c\ (= \mathbf{q_c}(M))$ and $\mathbf{h}_v^l(M)$ to generate parameters for our posterior distributions $q_\phi(\mathbf{z}_c|G)$ and $q_\phi(\mathbf{z}_l(j)|G),\ \forall j \in \{1 \dots |V|\}$.

Aggregation and Regularization Based Decoding

We propose a novel decoder different from the local proximity emphasized GVAE decoder to enforce the two qualities our common latent factor $\mathbf{z}_c$ should possess; $\mathbf{z}_c$ must be common for all patches and it should be relevant to input graph G. We utilize two decoders to fulfil these requirements.

Aggregation decoder $\mathcal{D}_{agg}$ to enforce commonality of $\mathbf{z}_c$ for all patches v. To reconstruct the original graph properly, the model requires both common and local factors. Therefore, common and local latent factors are sampled from their respective posterior distributions ($\mathbf{z}_c \sim q_\phi(\mathbf{z}_c|G)$ and $\mathbf{z}_l(j) \sim q_\phi(\mathbf{z}_l(j)|G),\ \forall j \in \{1 \dots |V|\}$) and sent through the decoder $\mathcal{D}_{agg}$ for reconstructing G. Note that the graph-wise common latent factors $\mathbf{z}_c$ is only sampled once for the entire graph using $q_\phi(\mathbf{z}_c|G)$. The aggregated reconstruction of G is achieved via adjacency matrix reconstruction A, where $p_\theta(A_{jk} = 1|\mathbf{z}_c, \mathbf{z}_l(j), \mathbf{z}_l(k)) = \mathcal{D}_{agg}([\mathbf{z}_c, \mathbf{z}_l(j)])^T\ \mathcal{D}_{agg}([\mathbf{z}_c, \mathbf{z}_l(k)])$.

Although $\mathcal{D}_{agg}$ enforces that $\mathbf{z}_c$ should contain factors common to all patches to enable proper graph reconstruction, $\mathcal{D}_{agg}$ cannot enforce what type of common factors $\mathbf{z}_c$ should possess. Since $\mathbf{z}_c$ is a constant for all patches, if the type of information that $\mathbf{z}_c$ should have was not enforced, there is a possibility that $\mathbf{z}_c$ gets ignored by $\mathcal{D}_{agg}$. To mitigate this, GCFX employs a **Regularization decoder** $\mathcal{D}_{reg}$ which enforces $\mathbf{z}_c$ that it must contain structural information about the graph G by trying to reconstruct G on its own as shown in Fig. 1.

Training and Inference Details

We modify the objective function in Eq. 6 obtained in the general framework for common latent factor extraction to following $\mathcal{L}_{deepGCFX}$ for training deepGCFX in an end-to-end manner.

$$\begin{aligned}\mathcal{L}_{deepGCFX} &= \mathcal{L}_{\mathcal{D}_{agg}} + \beta\mathcal{L}_{c\_prior} + \gamma\mathcal{L}_{l\_prior} + \mathcal{L}_{\mathcal{D}_{reg}} \\ &= \mathbb{E}_{q_\phi(\mathbf{z}_c, \mathbf{Z}_l|G)}[\log p_\theta(G|\mathbf{z}_c, \mathbf{Z}_c)] \\ &- \beta\ KL(q_\phi(\mathbf{z}_c|G)\ \|\ p(\mathbf{z}_c)) \\ &- \gamma \sum_{j=1}^{|V|} KL(q_\phi(\mathbf{z}_l(j)|G)\ \|\ p(\mathbf{z}_l(j))) \\ &+ \mathbb{E}_{q_\phi(\mathbf{z}_c, \mathbf{Z}_l|G)}[\log p_\theta(G|\mathbf{Z}_c)]\end{aligned} \tag{17}$$

Inference. After training deepGCFX as above, we utilize $\mathbf{z}_c \sim q_\phi(\mathbf{z}_c|G)$ as learnt single common latent factor representation for graph G and $\mathbf{z}_l(j) \sim q_\phi(\mathbf{z}_l(j)|G),\ \forall j \in \{1 \dots |V|\}$ as local non-common node/patch-specific latent factor representations for downstream tasks.

Evaluation

Effectiveness Analysis of DeepGCFX for Graph-Wise Common Latent Factor Extraction

To evaluate the effectiveness of deepGCFX in extracting graph-wise common latent factors, we analyze how the correlation of extracted common latent $\mathbf{z}_c$ varies with local latent factors $\mathbf{Z}_l$ and the output from the aggregation decoder $\mathcal{D}_{agg}$. Following graph recovery methods such as UDR(Duan et al. 2020), we use Spearman's correlation for this. Fig. 3 shows how the correlation changes with the number of iterations for a sample graph from the MUTAG dataset. We start with the accumulation iteration 0 where the common and local latent filtering is done at random to show how well the proposed ACCUM functions against random filtering.

We can observe that with the increase of ACCUM iterations, the correlation between local factors $\mathbf{Z}_l$ and common factors $\mathbf{z}_c$ decreases (Fig. 3(a)) showing our proposed ACCUM method's ability in feature differentiation by being able to filter different information as common and local latent factors. In Fig. 3(b) the correlation of $\mathbf{z}_c$ with the output of $\mathcal{D}_{agg}$ increases showing that $\mathcal{D}_{agg}$ considers common graph-level factors without overemphasizing on local proximity, hence ensuring commonality of $\mathbf{z}_c$. This shows the effectiveness of iterative ACCUM in extracting common factors.

In Fig. 4(a) we compare deepGCFX's filtering ability of $\mathbf{z}_c$ from $\mathbf{z}_l$ against GVAE which does not have that ability. To obtain two latent representations from GVAE, we divide

DATASET	MUTAG	PTC-MR	IMDB-BIN	IMDB-MUL	RED-BIN	RED-MUL-5K
Explicit inter-graph similarity based methods						
DDGK	91.6 ± 6.8	63.1 ± 6.6	–	–	–	–
GCKN-walk	**92.8 ± 6.1**	65.9 ± 2.0	**75.9 ± 3.7**	**53.4 ± 4.7**	–	–
UGraphEmb	-	**72.5**	-	50.1	–	–
Non-Explicit inter-graph similarity based methods						
Skip-gram based methods						
node2vec	72.6 ± 10.2	58.6 ± 8.0	–	–	–	-
sub2vec	61.1 ± 15.8	60.0 ± 6.4	55.3 ± 1.5	36.7 ± 0.8	71.5 ± 0.4	36.7 ± 0.4
graph2vec	83.2 ± 9.6	60.2 ± 6.9	71.1 ± 0.5	50.4 ± 0.9	75.8 ± 1.0	47.9 ± 0.3
Contrastive Learning based methods						
InfoGraph	89.0 ± 1.1	61.7 ± 1.4	73.0 ± 0.9	49.7 ± 0.5	82.5 ± 1.4	53.5 ± 1.0
CMV	89.7 ± 1.1	62.5 ± 1.7	74.2 ± 0.7	51.2 ± 0.5	84.5 ± 0.6	–
GCC	–	–	72.0	49.4	89.8	53.7
GraphCL	86.8 ± 1.3	–	71.1 ± 0.4	–	89.5 ± 0.4	**56.0 ± 0.3**
GVAE based methods						
GVAE(baseline)	87.7 ± 0.7	61.2± 1.8	70.7 ± 0.7	49.3 ± 0.4	87.1 ± 0.1	52.8 ± 0.2
deepGCFX (Ours) - α value for best results is in the brackets						
deepGCFX	89.8 ± 1.1	66.5 ± 1.0	72.9 ± 0.4	51.1±0.5	89.7±0.4	54.1±0.2
deepGCFX++	**92.2 ±0.9(0.7)**	**69.6± 1.4(0.85)**	**74.4± 0.2 (0.95)**	**52.7± 0.4(0.85)**	**90.9± 0.3(0.9)**	55.1± 0.2(0.85)

Table 1: Mean 10-fold cross validation accuracy on graph classification. Results in Bold indicate the best accuracy for both inter-graph similarity based and non-inter-graph similarity based methods separately. Underlined results show the second best performances.We follow strictly the experiment and evaluation setup and datasets as in (Sun et al. 2020; Hassani and Khasahmadi 2020) for deepGCFX and GVAE baseline. Results of other methods are taken from their papers.

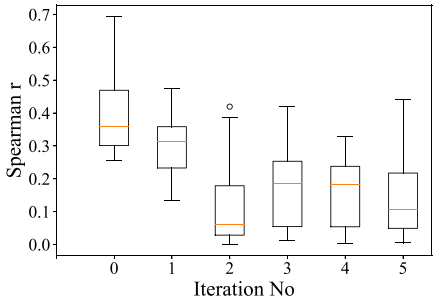
	ASSORTATIVE			DISASSORTATIVE		
DATASETS	CORA	CITESEER	PUBMED	CHAMELEON	SQUIRREL	ACTOR
Supervised Reference						
GCN	**85.77**	73.68	88.13	28.18	23.96	26.86
Geom-GCN	85.27	**77.99**	**90.05**	**60.90**	**38.14**	**31.63**
Unsupervised baselines						
DGI	**82.16 ±1.2**	**67.01± 1.3**	**81.34±0.6**	59.45±2.4	36.33±1.2	27.09± 1.2
GVAE	78.22 ±1.4	63.9 ±1.6	77.5± 0.7	56.88±2.9	33.05±1.6	25.12 ±1.4
deepGCFX - Ours						
$\mathbf{z}_c$	30.33± 1.2	20.75± 1.1	39.82± 0.5	19.30±2.7	19.23±0.8	10.5±1.2
$\mathbf{Z}_l$	81.26 ±1.2	65.51 ±1.4	79.85± 0.7	57.67±3.1	35.64±1.7	26.88± 1.0
deepGCFX++	81.96 ±1.7(0.15)	66.71±1.6(0.1)	80.3± 0.7(0.2)	**61.05±2.4**(0.35)	**39.20±1.4**(0.4)	**28.80± 1.4**(0.4)

Table 2: Mean node classification accuracy for supervised and unsupervised models for assortative and disassortative graphs. Results in Bold indicate best supervised and unsupervised accuracy for each dataset and underlined is the second best for unsupervised. We strictly follow the evaluation setup and datasets of Geom-GCN(Pei et al. 2020). For unsupervised methods we used linear evaluation protocol.

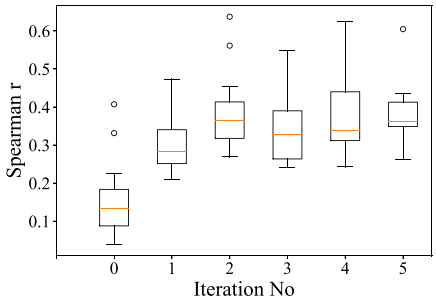
the learnt latent into halves. We obtain Spearman R based correlation matrix and by analysing them we can conclude that, deepGCFX indeed can differentiate features, as the correlation between $\mathbf{z}_c$ from $\mathbf{z}_l$ is very low compared to GVAE. To verify the factors extracted by our query-based reasoning ACCUM indeed are common across the graph, we use the experiment in Fig. 4(b). We use Mean Absolute Pairwise Difference (MAPD) measure used by Higgins et al. (2017) to compare the inter-patch similarity of common and local latent separately and we can observe that inter-patch common latent factor similarity is very high compared to inter-patch local factors demonstrating that the latent factors extracted by our ACCUM method are indeed common across the graph.

Impact of Learnt Graph-Wise Common Factors $\mathbf{z}_c$ and Local Factors $\mathbf{z}_l$ on Downstream Task Performance.

To evaluate the discriminative ability of extracted common latent factors $\mathbf{z}_c$ on downstream tasks, we select graph classification. We report results for deepGCFX when only $\mathbf{z}_c$ is used as the graph embedding. deepGCFX++ combines both common and local factors with a gating mechanism as $\alpha\mathbf{z}_c + (1-\alpha)\sum_{j=1}^{|V|}\mathbf{z}_l(j)$, where α denotes the contribution from graph-wise common factors. We compare deepGCFX with existing state-of-the-art methods and report results in Table 1. Compared to existing work on skip-gram and contrastive learning, deepGCFX achieves comparable or better results for four datasets and our deepGCFX++(when

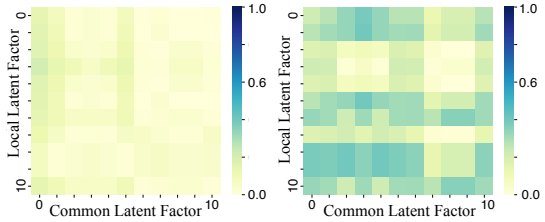


(a) Correlation between $\mathbf{z}_c$ and $\mathbf{Z}_l$

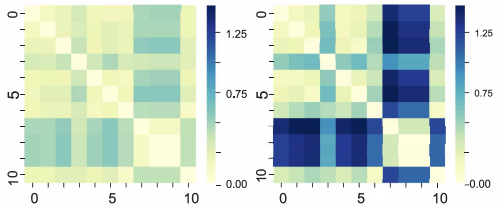


(b) Correlation between $\mathbf{z}_c$ and output from $\mathcal{D}_{agg}$

Figure 3: Variation of Spearman R absolute correlation of extracted common latent factors $\mathbf{z}_c$ with local ($\mathbf{Z}_l$) and the output from $\mathcal{D}_{agg}$ against number ACCUM iterations.



(a) Correlation between common and local factors from deepGCFX (left) vs GVAE (right)



(b) Patch-wise MAPD for Common(left) and Local(right) factors

Figure 4: (a) Filtering ability of learnt Common and Local latent factors by deepGCFX via absolute values of correlation compared with GVAE, which does not perform common-local filtering. (b) Inter-patch MAPD among Common and Local latent factors. Lower MAPD for $\mathbf{z}_c$ indicates the common factor representations filtered by deepGCFX is indeed shared across patches of the entire graph, unlike $\mathbf{Z}_l$ which are specific to certain patches.

the biggest contribution comes from common latent factors) achieves the state-of-the-art results for 5 datasets and is very competitive with GraphCL(You et al. 2020) which uses data augmentations for REDDI-MULTI-5K showing the effectiveness of GCFX over contrastive learning, whose model performance relies on the selection of negative samples. We achieve very competitive results with inter-graph similarity methods by only using current graph for embedding learning, compared to their pair-wise comparisons. These results show the effectiveness of utilizing graph-wise common factors as graph embeddings.

Impact of $\mathbf{z}_c$ for Node-Level Tasks

To analyze how common latent factors affect node-level tasks, we select node classification task on both assortative (neighbour nodes of a graph have same class label) and disassortative (neighbour nodes have different class labels) graph. Since $\mathbf{z}_c$ is common for all the nodes in the graph despite the inter-node distance, we want to analyze whether it is beneficial in improving long-range node dependencies. Non-local aggregating (Pei et al. 2020; Liu, Wang, and Ji 2020) for graph learning has drawn attention in supervised graph learning research due to GNN's inability in long-distance information propagation. Geom-GCN (Pei et al. 2020) proposed a benchmark to evaluate non-local aggregation for both assortative and disassortative graphs.

To the best of our knowledge, existing unsupervised graph representation learning methods have not used this benchmark for evaluation, hence we select two supervised (GCN(Kipf and Welling 2017), Geom-GCN(Pei et al. 2020)) for the reference and two unsupervised (Deep Graph Infomax (DGI)(Velickovic et al. 2019), GVAE(Kipf and Welling 2016)) methods as our baselines. Table 2 reports results for all models and, compared to local only ($\mathbf{Z}_l$) of deepGCFX, deepGCFX++ which incorporates common latent factors with local factors achieves higher results demonstrating the effectiveness of common latent factors for node-level tasks. More interestingly, deepGCFX++ has achieved best results for disassortative graphs highlighting extracted common latent factors' ability in capturing long-distance node dependencies.

Conclusion

We introduced a graph-wise common latent factor extraction based principle for unsupervised graph representation learning. Based on real-world graphs, we identified common factors that are used along with node-specific local factors for graph generation and we hypothesized extracting common factors could be highly beneficial for discriminative graph representations. We proposed GCFX principle and deepGCFX model to address feature differentiation and proximity overemphasis limitations of GVAE to enable graph-wise common latent factor extraction. With extensive experiments, we demonstrated the effectiveness of deepGCFX and its outstanding performance over state-of-the-art methods with only using current sample for embedding learning.

Acknowledgments

This work was supported by SUTD project PIE-SGP-AI-2018-01. This research was also supported by the National Research Foundation Singapore under its AI Singapore Programme [Award Number:AISG-100E2018-005]. The authors are grateful for the discussion with Lu Wei.

References

Adhikari, B.; Zhang, Y.; Ramakrishnan, N.; and Prakash, B. A. 2018. Sub2Vec: Feature Learning for Subgraphs. In *Advances in Knowledge Discovery and Data Mining - 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3-6, 2018, Proceedings, Part II*, 170–182.

Al-Rfou, R.; Perozzi, B.; and Zelle, D. 2019. DDGK: Learning Graph Representations for Deep Divergence Graph Kernels. In Liu, L.; White, R. W.; Mantrach, A.; Silvestri, F.; McAuley, J. J.; Baeza-Yates, R.; and Zia, L., eds., *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, 37–48. ACM.

Bai, Y.; Ding, H.; Qiao, Y.; Marinovic, A.; Gu, K.; Chen, T.; Sun, Y.; and Wang, W. 2019. Unsupervised Inductive Graph-Level Representation Learning via Graph-Graph Proximity. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 1988–1994.

Baldi, P.; and Hornik, K. 1989. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2: 53–58.

Borgwardt, K. M.; and Kriegel, H. P. 2005. Shortest-path kernels on graphs. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, 8 pp.–.

Brown, N.; Fiscato, M.; Segler, M. H.; and Vaucher, A. C. 2019. GuacaMol: Benchmarking Models for de Novo Molecular Design. *Journal of Chemical Information and Modeling*, 59(3): 1096–1108.

Chen, D.; Jacob, L.; and Mairal, J. 2020. Convolutional Kernel Networks for Graph-Structured Data. *CoRR*, abs/2003.05189.

Cooray, T.; Cheung, N.-M.; and Lu, W. 2020. Attention-Based Context Aware Reasoning for Situation Recognition. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Duan, S.; Matthey, L.; Saraiva, A.; Watters, N.; Burgess, C.; Lerchner, A.; and Higgins, I. 2020. Unsupervised Model Selection for Variational Disentangled Representation Learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Duvenaud, D. K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; and Adams, R. P. 2015. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28*, 2224–2232. Curran Associates, Inc.

Gärtner, T.; Flach, P. A.; and Wrobel, S. 2003. On Graph Kernels: Hardness Results and Efficient Alternatives. In *Computational Learning Theory and Kernel Machines, 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003, Proceedings*, 129–143.

Grill, J.; Strub, F.; Altché, F.; Tallec, C.; Richemond, P. H.; Buchatskaya, E.; Doersch, C.; Pires, B. Á.; Guo, Z.; Azar, M. G.; Piot, B.; Kavukcuoglu, K.; Munos, R.; and Valko, M. 2020. Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, 855–864.

Hassani, K.; and Khasahmadi, A. H. 2020. Contrastive Multi-View Representation Learning on Graphs. In *Proceedings of International Conference on Machine Learning*, 3451–3461.

Higgins, I.; " i c Matthey, L.; Pal, A.; Burgess, C.; Glorot, X.; Botvinick, M.; Mohamed, S.; and Lerchner, A. 2017. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Hinton, G. E.; and Zemel, R. S. 1993. Autoencoders, Minimum Description Length and Helmholtz Free Energy. In *Advances in Neural Information Processing Systems 6, [7th NIPS Conference, Denver, Colorado, USA, 1993]*, 3–10.

Kipf, T. N.; and Welling, M. 2016. Variational Graph Auto-Encoders. *CoRR*, abs/1611.07308.

Kipf, T. N.; and Welling, M. 2017. 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*.

Krishna, R.; Zhu, Y.; Groth, O.; Johnson, J.; Hata, K.; Kravitz, J.; Chen, S.; Kalantidis, Y.; Li, L.; Shamma, D. A.; Bernstein, M. S.; and Fei-Fei, L. 2017. Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations. *Int. J. Comput. Vis.*, 123(1): 32–73.

Krogan, N.; Cagney, G.; Yu, H.; Zhong, G.; Guo, X.; Ignatchenko, A.; Li, J.; Pu, S.; Datta, N.; Tikuisis, A.; Punna, T.; Peregrín-Alvarez, J.; Shales, M.; Zhang, X.; Davey, M.; Robinson, M.; Paccanaro, A.; Bray, J.; Sheung, A.; and Greenblatt, J. 2006. Global landscape of protein complexes in the yeast Saccharomyces cerevisiae. *Nature*, 440: 637–43.

Kuhn, C.; and Beratan, D. N. 1996. Inverse strategies for molecular design. *The Journal of Physical Chemistry*, 100(25): 10595–10599.

Linsker, R. 1988. Self-Organization in a Perceptual Network. *computer*, 21(3): 105–117.

Liu, M.; Wang, Z.; and Ji, S. 2020. Non-local graph neural networks. *arXiv preprint arXiv:2005.14612*.

Liu, R.; and Cheung, N. 2021. Joint estimation of low-rank components and connectivity graph in high-dimensional graph signals: Application to brain imaging. *Signal Process.*, 182: 107931.

Narayanan, A.; Chandramohan, M.; Venkatesan, R.; Chen, L.; Liu, Y.; and Jaiswal, S. 2017. graph2vec: Learning Distributed Representations of Graphs. *CoRR*, abs/1707.05005.

Newman, M. E. J.; and Girvan, M. 2004. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69: 026113.

Pan, S.; Hu, R.; Long, G.; Jiang, J.; Yao, L.; and Zhang, C. 2018. Adversarially Regularized Graph Autoencoder for Graph Embedding. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, 2609–2615.

Pang, J.; and Cheung, G. 2017. Graph Laplacian Regularization for Image Denoising: Analysis in the Continuous Domain. *IEEE Transactions on Image Processing*, 26(4): 1770–1785.

Park, J.; Lee, M.; Chang, H. J.; Lee, K.; and Choi, J. Y. 2019. Symmetric Graph Convolutional Autoencoder for Unsupervised Graph Representation Learning. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, 6518–6527.

Pei, H.; Wei, B.; Chang, K. C.; Lei, Y.; and Yang, B. 2020. Geom-GCN: Geometric Graph Convolutional Networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Qiu, J.; Chen, Q.; Dong, Y.; Zhang, J.; Yang, H.; Ding, M.; Wang, K.; and Tang, J. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. *arXiv preprint arXiv:2006.09963*.

Sanchez-Lengeling, B.; and Aspuru-Guzik, A. 2018. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400): 360–365.

Schneider, G. 2013. *De novo molecular design*. John Wiley & Sons.

Shervashidze, N.; Vishwanathan, S. V. N.; Petri, T.; Mehlhorn, K.; and Borgwardt, K. M. 2009. Efficient graphlet kernels for large graph comparison. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009*, 488–495.

Sun, F.; Hoffmann, J.; Verma, V.; and Tang, J. 2020. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Tian, Y.; Krishnan, D.; and Isola, P. 2020. Contrastive Multiview Coding. In Vedaldi, A.; Bischof, H.; Brox, T.; and Frahm, J., eds., *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XI*, volume 12356 of *Lecture Notes in Computer Science*, 776–794. Springer.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; ‘o, P. L.; and Bengio, Y. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Velickovic, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2019. Deep Graph Infomax. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Wieder, O.; Kohlbacher, S.; Kuenemann, M.; Garon, A.; Ducrot, P.; Seidel, T.; and Langer, T. 2020. A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Yada, K.; Motoda, H.; Washio, T.; and Miyawaki, A. 2004. Consumer Behavior Analysis by Graph Mining Technique. In Negoita, M. G.; Howlett, R. J.; and Jain, L. C., eds., *Knowledge-Based Intelligent Information and Engineering Systems, 8th International Conference, KES 2004, Wellington, New Zealand, September 20-25, 2004. Proceedings. Part II*, volume 3214 of *Lecture Notes in Computer Science*, 800–806. Springer.

Yanardag, P.; and Vishwanathan, S. V. N. 2015. Deep Graph Kernels. In Cao, L.; Zhang, C.; Joachims, T.; Webb, G. I.; Margineantu, D. D.; and Williams, G., eds., *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, 1365–1374. ACM.

Yang, K.; Swanson, K.; Jin, W.; Coley, C.; Eiden, P.; Gao, H.; Guzman-Perez, A.; Hopper, T.; Kelley, B.; Mathea, M.; et al. 2019. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8): 3370–3388.

You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020. Graph Contrastive Learning with Augmentations. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Zunger, A. 2018. Inverse design in search of materials with target functionalities. *Nature Reviews Chemistry*, 2(4): 1–16.